



HAL
open science

Logic-based Cognitive Planning: from theory to implementation

Jorge Luis Fernandez Davila

► **To cite this version:**

Jorge Luis Fernandez Davila. Logic-based Cognitive Planning: from theory to implementation. Artificial Intelligence [cs.AI]. Université Paul Sabatier - Toulouse III, 2022. English. NNT: 2022TOU30193 . tel-04136582

HAL Id: tel-04136582

<https://theses.hal.science/tel-04136582>

Submitted on 21 Jun 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Fédérale



Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE
Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par
Jorge LUIS FERNANDEZ DAVILA

Le 23 septembre 2022

**Planification cognitive basée sur la logique: de la théorie à
l'implémentation**

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et
Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :

IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Emiliano LORINI

Jury

M. Bruno ZANUTTINI, Rapporteur
M. Tiago DE LIMA, Rapporteur
Mme Serena VILLATA, Examinatrice
Mme Sylvie THIEBAUX, Examinatrice
Mme Leila AMGOUD, Examinatrice
M. Dominique LONGIN, Examineur
M. Emiliano LORINI, Directeur de thèse

Abstract

In this thesis, we define a logic-based cognitive planning framework for endowing artificial agents with the skills aimed to persuade humans to believe something or to induce a certain behavior in them. Our cognitive planning framework is based on an NP-complete fragment of an epistemic logic with a semantics exploiting belief bases and whose satisfiability problem can be reduced to SAT. We propose a general architecture for the cognitive planning problem, which considers two main components: the belief revision and the cognitive planning module. We formalized two types of planning problem: informative and interrogative, and we studied the complexity of finding a solution in both cases. Moreover, we introduce an alternative encoding to the SAT approach for finding a solution to the informative problem using quantified boolean formula (QBF), which considers an optimal number of quantifiers in the prefix. We illustrate the potential of our framework for cognitive planning in the context of HMI by implementing two examples. In the first example, an artificial assistant recommends to a human user an ideal sport to practice based on the human's preferences. In the second example, we extend our epistemic language to represent the beliefs and actions of an artificial player in the context of the cooperative board game Yokai. This game requires a combination of the theory of mind, temporal reasoning, and spatial reasoning for an artificial agent to play effectively. Our implementations demonstrate that our NP-complete fragment and the cognitive planning problem formulated in this logic are suitable for real-world applications in the domain of HMI.

Résumé

Dans cette thèse, nous définissons un cadre de planification cognitive basé sur la logique pour doter les agents artificiels des compétences visant à persuader les humains de croire quelque chose ou d'induire un certain comportement en eux. Notre cadre de planification cognitive est basé sur un fragment NP-complet d'une logique épistémique avec une sémantique exploitant des bases de croyances et dont le problème de satisfiabilité peut être réduit à SAT. Nous proposons une architecture générale pour le problème de planification cognitive, qui considère deux composantes principales: la révision des croyances et le module de planification cognitive. Nous avons formalisé deux types de problème de planification: informatif et interrogatif, et nous avons étudié la complexité de trouver une solution dans les deux cas. De plus, nous introduisons un codage alternatif à l'approche SAT pour trouver une solution au problème informatif en utilisant une formule booléenne quantifiée (QBF), qui considère un nombre optimal de quantificateurs dans le préfixe. Nous illustrons le potentiel de notre cadre de planification cognitive dans le contexte de l'HMI en mettant en œuvre deux exemples. Dans le premier, un assistant artificiel recommande à un utilisateur humain un sport idéal à pratiquer en fonction des préférences de l'humain. Dans le deuxième exemple, nous étendons notre langage épistémique pour représenter les croyances et les actions d'un joueur artificiel dans le contexte du jeu de société coopératif Yokai. Ce jeu nécessite une combinaison de la théorie de l'esprit, du raisonnement temporel et du raisonnement spatial pour qu'un agent artificiel joue efficacement. Nos implémentations démontrent que notre fragment NP-complet et le problème de planification cognitive formulé dans cette logique conviennent aux applications du monde réel dans le domaine de l'HMI.

Acknowledgments

First and foremost, I would like to express my gratitude to my thesis supervisor Emiliano Lorini and my co-supervisors, Dominique Login and Frédéric Maris. Thanks to Emiliano for always listening to me, understanding what I had issues with, and guiding me in a direction where I could “comfortably” excel. This was because he knew what my strengths and weaknesses were. He never gave up on me and always preserved his confidence in my ability to complete this Ph.D., but overall thank you for always challenging me to probe myself. Also, I owe a lot from these three years to Dominique, co-author in several of these investigations. I wish to thank Dominique for helping me to ground the philosophical concepts and relate them to my engineering background. Our meetings always involved funny stories related to what we dealt with, especially when we worked on the Yokai game’s modelization, which involved countless hours of intensive meetings and tests that were finally reflected in Chapter 5. Similarly, my gratitude to Frédéric, whose precise and concise advice led us to implement the QBF version of our planning algorithm in a few weeks. Thanks to that, we could devote more time to the experimental part when we compare the QBF version and the brute force approach of our cognitive planning algorithm, outcomes that are detailed in Chapter 3.

I also extend my deepest gratitude to Andreas Herzig, who gave us insightful advice during the definition of the translations from our logic to propositional logic, which later would become the main set of reductions in our logical framework mentioned in Chapter 2. I wish to acknowledge the help provided by the DAVI company technical staff. Thanks to Yannick Gérard, Delphine Potdevin, Francisco Sanchez, and Raphaël Zaragoza for their continuous support, which allowed us to successfully integrate the emotional AI avatar interface into our cognitive planning architecture, work that is described in Chapter 4. People who came to IRIT to give lectures and with whom I had the opportunity to talk about the topics of this thesis, Carlos Areces, who dedicated part of his time to provide me with an overview of similar works in this field of research. Thanks to Renata Weissman for her valuable comments concerning the belief revision part. My appreciation goes equally to Antonio Yuste Ginel for his advice on the logical framework.

I wish to express my most sincere gratitude and appreciation to my thesis committee members: Leila Amgoud, Bruno Zanuttini, Tiago de Lima, Serena Villata, Sylvie Thiébaux and Elise Perrotin, whose comments and questions during the defense gave me a chance to reflect on possible new branches of this research. Special thanks to Tiago and Bruno for their careful reading of this thesis and whose detailed reports that help me to improve it significantly.

Undertaking a Ph.D. has been one of the best decisions in my life, although it has been a challenging road. However, I am grateful to have had the opportunity to light my journey by sharing moments with my colleagues Thorsten Engesser, Mynque Mittelmann, Julien Vianey, Saúl Fernández González, Xinghan Liu and Mouna Mayouf, whose hard work and dedication to research motivated me to continue and overcome the challenges.

Thank God, my parents, my wife and my children. It would have been impossible to finish my studies without their unwavering support over the past few years.

To conclude, I would like to thank the ANR for funding this thesis work through the CoPains PRCE-Défi 7-Axe 3-2018 project (<https://www.irit.fr/CoPains/>).

Contents

Introduction	1
Research questions	2
Motivation and contribution of the thesis	3
Outline of the thesis	3
Sources of the chapters	4
1 Basics	5
1.1 Epistemic Logic	5
1.2 Doxastic logic	9
1.3 Dynamic Epistemic Logic - DEL	9
1.4 Logic of Doxastic Attitudes - LDA	12
1.5 Epistemic planning	13
1.6 Satisfiability - SAT	13
1.7 Quantified Boolean Formula - QBF	15
2 Logical Framework	17
2.1 Full Language and Semantics	17
2.2 NP-Complete Fragment	19
2.3 Dynamic Extension	23
2.4 Conclusion	25
3 Cognitive Planning	27
3.1 Related Work on Epistemic Planning and Persuasion	27
3.2 General Architecture	29
3.3 Planning Problems	30
3.4 Complexity results	34
3.5 Belief Revision Module	36
3.6 Example 1: Artificial assistant	38
3.7 Example 2: Virtual coaching agent	43
3.7.1 Motivational interviewing	43
3.7.2 Formalization	44
3.8 Optimal QBF Encoding	48
3.9 Conclusion	53
4 An Implemented System for Cognitive Planning	55
4.1 Implementation	55
4.2 Experiments	65
4.3 Discussion	68
4.4 Conclusion	69

5	A Logical Modeling of the Yōkai Board Game	71
5.1	Introduction	71
5.2	Game description	73
5.3	A timed language for explicit and implicit belief	75
5.3.1	Static language	75
5.3.2	Dynamic extension	78
5.4	Artificial agent architecture	80
5.4.1	Action selection	80
5.4.2	Belief change	80
5.5	Game modeling	84
5.5.1	Static aspects	85
5.5.2	Dynamic aspects	90
5.5.3	Example of action selection	92
5.6	Goals modeling	92
5.6.1	Observe actions	94
5.6.2	Move actions	97
5.6.3	Mark actions and active actions	100
5.7	Implementation and experiments	102
5.8	Discussion	107
5.9	Conclusion	108
	Conclusion	109
	Appendices	111
	Appendix A	113
	Detailed proof of Theorem 10	113
	Detailed proof of Theorem 11	117
	The separation constraint	118
	Appendix B	121
	Implementation of Yōkai	121
	Grouping cards	132
	Action selection	136
	Belief revision	138
	Hierarchy of goals	140
	Bibliography	143

Introduction

Automated planning is at the center of AI research with a variety of applications ranging from control traffic and robotics to logistics and services. Epistemic planning extends automated planning by incorporating notions of knowledge and beliefs [Bolander & Andersen 2011, Löwe *et al.* 2011, Kominis & Geffner 2015, Muise *et al.* 2015, Cooper *et al.* 2021]. Cognitive planning is a generalization of epistemic planning, where the goal to be achieved is not only a belief state but a cognitive state of a target including not only beliefs but also intentions. Moreover, we are particularly interested in the planning agent’s persuasive goals, aimed at influencing another agent’s beliefs and intentions.

In this thesis, an integrated framework for cognitive planning is proposed. A key aspect of our framework is that it is based on an epistemic logic that adopts the “database perspective” [Lorini 2018, Lorini 2020]. This feature allows us to model agents’ “mental states” in a compact way. Furthermore, we develop an automated reasoning procedure for this language based on an extension of an encoding tool called TouIST (Toulouse Integrated Satisfiability Tool) [Fernandez *et al.* 2020], which provides a flexible and intuitive syntax for writing logical formulas that allows us to encode agents’ cognitive states.

In the subsequent stage, the machine will reason about the facts represented in its database to deduce information about the human agent’s beliefs and intentions. In other words, the machine can predict human behavior based on its model of the human’s mind, and given a goal (persuasive or influential), the machine will select a sequence of speech acts aimed at changing the human agent’s behavior.

We demonstrated through an implementation that the proposed architecture leads to the rise of a sufficiently viable artificial reasoning agent. The system architecture considers two core components, the belief revision and the cognitive planning module. For the belief revision module, we use maximal consistent subsets. We consider a brute force algorithm based on SAT (Boolean satisfiability problem) for the cognitive planning module. The implemented artificial agent interacts with the human to collect information and increases its ability to understand the human agent’s mental state. Based on experimental results, we conclude that it is possible to use our cognitive planning framework for implementing real-world applications. Furthermore, we proposed an optimal encoding for the cognitive planning problem, expressing the planning formula in terms of a quantified boolean formula (QBF). The experimental evaluation concludes that the QBF approach outperforms the brute force technique based on SAT for solving the cognitive planning problem.

Finally, we applied our cognitive planning framework to a case of a cooperative board game called Yōkai, involving the human and the machine in which they have to exchange information and collaborate in order to achieve a common goal. Yōkai requires a combination of Theory of Mind (ToM), temporal and spatial reasoning to be played effectively by the artificial agent. We showed that the language properly

accounts for these three dimensions and proved that its satisfiability problem is NP-complete. Moreover, we implement *Yōkai* and we perform experiments to evaluate the performance between two game configurations: human-machine versus human-human collaboration.

Research questions

We want to find the answers to four fundamental research questions in our path to building a cognitive planning framework. We can formulate them as follows:

1. Can we build a logical framework whose satisfiability problem can be reduced to SAT to represent and reason about an agent's beliefs, desires, and intentions?

To answer this question, we need to study a multi-agent epistemic logic that allows us to represent and reason about other agents' beliefs and intentions and whose complexity allows us to move towards a realistic implementation. Moreover, we are expecting to use SAT solvers, so we need to define a sequence of reductions to transform formulas expressed in our logical language to their equivalent propositional logic version.

2. Can we propose an integrated architecture for cognitive planning in the context of our logical framework?

To answer this, we need first to define the cognitive planning problem and analyze the complexity of finding a solution plan. Moreover, we need to consider two types of planning problem: informative and interrogative. Finally, if we are expecting to model a dialog between the human and the machine, it is necessary to capture the input coming from the human using a belief revision technique.

3. How can we implement the integrated architecture for cognitive planning using functional programming?

We need to identify a strong mathematical language suitable for implementing the recursive functions that will allow us to transform the planning formulas into their propositional logic version. In addition, we need to select a SAT solver tool to verify the validity of the planning formulas expressed in propositional logic.

4. Is it possible to apply our cognitive planning model for the case of a cooperative game that uses ToM as well as temporal and spatial reasoning?

To answer this question we need to extend the language to include the temporal aspects. In addition, to cooperate with humans the machine should be able to deduce information given by the human agent by means of hints.

Motivation and contribution of the thesis

Our motivation is to endow an artificial agent with cognitive planning capabilities in order to persuade humans to believe something or behave in a certain way. In order to do that, we are expecting to build a model of the human agent's mind from the daily interaction between the machine and the human. This model will contain the interconnection between the human's beliefs, desires, and intentions. In addition, we will set up a persuasive or influential goal and a set of actions in the machine.

These actions will be in the form of speech acts. During the cognitive planning process, the machine will choose a sequence of speech acts from the set of actions that will lead the machine to reach the goal. In other words, the cognitive planning process consists of identifying a sequence of speech acts that will be successful in persuading the human agent.

Our main contribution is the development of an integrated framework for cognitive planning. Our work goes towards the integration effort, bringing together three fields related to our approach: Dynamic Epistemic Logic, Knowledge Representation and Reasoning, and Automated Reasoning unifying these fields in an integrated framework for cognitive planning. We believe that our logical framework and architecture are general enough and technology agnostic to be implemented in different tools. In order to show its potential for application, we implement our model and instantiate it in a scenario of human machine interaction (HMI).

Outline of the thesis

In this section we provide a short summary of each chapter.

In Chapter 1, we give an introduction to the different fields of research this thesis contributes to, starting with a list of logics for which we summarize the standard syntax and semantics: Epistemic Logic, Doxastic Logic, Dynamic Epistemic Logic and the Logic of Doxastic Attitudes. We also illustrate, with an example, the fundamentals of Epistemic Planning. The last two sections of the first chapter then introduce the Satisfiability and Quantified Boolean formula problems.

In Chapter 2, an NP-complete fragment of an epistemic logic with a semantics exploiting belief bases is presented. We start by recalling the syntax and the semantics for the full logic. Then, we define our NP-complete fragment and its dynamic extension. Next, we detail the set of translations for the reduction of our fragment to SAT. Finally, we provide complexity results for checking satisfiability of formulas in our NP-fragment.

In Chapter 3, we define a general architecture for the cognitive planning problem. Afterward, we define two types of planning problem: informative and interrogative, and we find the complexity of finding a solution for the cognitive planning problem in both cases. Furthermore, we illustrated the potential of our framework for applications in human-machine interaction with the help of two examples in which an artificial agent is expected to interact with a human agent through dialogue and to

persuade the human to behave in a certain way. At the end of this chapter we introduced a formalization of simple cognitive planning as a quantified boolean formula (QBF) with an optimal number of quantifiers in the prefix.

The model for cognitive planning is implemented in Chapter 4. We describe how to represent and generate the belief base. Furthermore, we demonstrate how the machine performs the reasoning process to find a sequence of speech acts intended to induce a potential intention in the human agent. The implemented system has three main components: belief revision, cognitive planning, and the translator module. These modules work integrated to capture the human agent's beliefs during the human-machine interaction process and generate a sequence of speech acts to achieve a persuasive goal.

In Chapter 5, we present an epistemic language to represent the beliefs and actions of an artificial player in the context of the board game *Yōkai*. The cooperative game *Yōkai* requires a combination of theory of mind (ToM), temporal and spatial reasoning for an artificial agent to play effectively. We show that the language properly accounts for these three dimensions, and we show that its satisfiability problem is NP-complete. We implement the game and perform experiments to compare the level of cooperation in two scenarios: when the game is played between a human and the artificial agent versus when two humans play the game.

Sources of the chapters

Parts of this thesis have been published before. Below, we give a quick overview of the publications and ongoing works on which each chapter is based.

- Chapter 1 is a new summary of basic concepts and ideas from the literature.
- Chapter 2 is based on : Fernandez Davila, J. L., Longin, D., Lorini, E., Maris, F. (2021). A Simple Framework for Cognitive Planning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(7), 6331-6339.
- Chapter 3 is based on : Lorini, E.; Sabouret, N.; Ravenet, B.; Fernandez, J. and Clavel, C. (2022). Cognitive Planning in Motivational Interviewing. In Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2, ISBN 978-989-758-547-0, ISSN 2184-433X, pages 508-517.
- Chapter 4 is based on : Fernandez, J.; Longin, D.; Lorini, E. and Maris, F. (2022). An Implemented System for Cognitive Planning. In Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3, ISBN 978-989-758-547-0, ISSN 2184-433X, pages 492-499.
- Chapter 5 is based on an ongoing work. Jorge Luis Fernandez Davila, Dominique Longin, Emiliano Lorini, Frédéric Maris. A Logical Modeling of the *Yōkai* Board Game. [Research Report] IRIT - Institut de recherche en informatique de Toulouse. 2020.

This thesis combines ideas from epistemic logic and computer science. In this preliminary chapter we introduce the basic building blocks of our framework. Depending on their background, the reader should feel free to skip over sections about structures or methods already known to them.

1.1 Epistemic Logic

The term epistemic logic (EL) can be understood in two senses. In the broader sense, it includes the analysis of both the concept of knowledge and the weaker concept of belief. This is the way in which it is used throughout this thesis. In a narrower sense, the name epistemic logic is reserved for the former, leaving for the latter the name of *doxastic logic* (from the Greek $\delta\acute{o}\xi\alpha$: belief). When it is not clear from the context, it will be specified which meaning is targeted.

Since epistemic logic is an extension of propositional logic (PL), we start by defining the syntax for a PL formula.

Definition 1 (Language of propositional logic) *Let Atm be a set of atomic propositions. The set of formulas in boolean propositional logic language, denoted by \mathcal{L}_{Prop} , is defined by the following grammar in BNF:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi,$$

where $p \in Atm$. In terms of the semantics, the meaning of \mathcal{L}_{Prop} formulas is given by an interpretation I that gives the truth value for each atom. Given an interpretation, we can assign values to \mathcal{L}_{Prop} formulas using the logical operators.

Definition 2 (Satisfaction) *The \mathcal{L}_{Prop} formula φ is true in an interpretation (an assignment) I , written $I \models \varphi$, as inductively defined by distinguishing the shape of formula φ . Then:*

$$\begin{aligned} I \models p &\iff I(p) = \text{true for atoms } p, \\ I \models \neg\varphi &\iff I \not\models \varphi, \\ I \models \varphi \wedge \psi &\iff I \models \varphi \text{ and } I \models \psi. \end{aligned}$$

In the *Theaetetus*, Plato argues that knowledge is true belief plus “something else” which is often difficult to define and which he described as *rational justification*. Thus, the following condition can be applied to knowledge: if agent i knows that φ , then φ is true. On the other hand, we shall argue that it is not possible to apply such a rule to the concept of belief. In fact, it is widely accepted that beliefs can be false. Therefore, when we are talking about *belief* we refer to a *rational belief*, i.e., agent i doesn’t believe $\varphi \wedge \neg\varphi$.

Epistemic logic studies what agents know or believe about certain facts. The following are some examples of expressions formulated by epistemic logic:

- (1) Agent i knows that φ
- (2) Agent i knows whether φ
- (3) Agent i does not know φ
- (4) Agent i does not know whether φ
- (5) It is possible, taking into account what agent i knows, that φ

In order to formalize these expressions, epistemic logic introduces two modal operators K and \widehat{K} , using a sub index to identify the agent who owns the knowledge:

- (1) $K_i\varphi$
- (2) $K_i\varphi \vee K_i\neg\varphi$
- (3) $\neg K_i\varphi$
- (4) $\neg K_i\varphi \wedge \neg K_i\neg\varphi$
- (5) $\widehat{K}_i\varphi$

Using Kripke’s terms, we will understand that expression (1) informs us that φ is true in all possible worlds compatible with what agent i knows, while (5) tells us that φ is true in at least one possible world compatible with what agent i knows. We define the dual operator $\widehat{K}_i\varphi$ as follows: $\widehat{K}_i\varphi \stackrel{\text{def}}{=} \neg K_i\neg\varphi$.

Definition 3 (Language of epistemic logic) Let $\text{Agt} = \{1, \dots, n\}$ a finite set of agents. The language $\mathcal{L}_K(\text{Atm}, \text{Agt})$, the language for multiagent epistemic logic, is an extension of $\mathcal{L}_{\text{Prop}}$ with modal operators for knowledge, represented by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi$$

where $p \in \text{Atm}$ and $i \in \text{Agt}$. We read $K_i\varphi$ as “agent i knows φ ”.

When Atm and Agt are clear from the context, we will write \mathcal{L}_K instead of $\mathcal{L}_K(\text{Atm}, \text{Agt})$.

Definition 4 (Kripke model) An epistemic model on (Atm, Agt) is a structure $M = \langle W, (R_i)_{i \in \text{Agt}}, V \rangle$, where :

- W is a non-empty set of possible worlds (or states),
- $R_i \subseteq W \times W$ represents the epistemic accessibility relation for agent $i \in \text{Agt}$.

- $V : Atm \rightarrow 2^W$ is a valuation function which associates to every $p \in Atm$ the set of worlds in which p is true.

We often suppress the reference to the agent in the accessibility relation and we simply write $M = \langle W, R, V \rangle$. The truth of formulas in \mathcal{L}_K are interpreted with respect to a pointed Kripke model. A pointed Kripke model is a pair (M, w) where w is a world of W .

Definition 5 (Satisfaction) Given a model $M = \langle W, R, V \rangle$ and a pointed Kripke model (M, w) . Then we can interpret formulas in \mathcal{L}_K as:

$$\begin{aligned} (M, w) \models p &\iff w \in V(p) , \\ (M, w) \models \neg\varphi &\iff (M, w) \not\models \varphi , \\ (M, w) \models (\varphi \wedge \psi) &\iff (M, w) \models \varphi \text{ and } (M, w) \models \psi, \\ (M, w) \models K_i\varphi &\iff (M, w') \models \varphi \text{ for all } w' \text{ such that } wR_iw' \end{aligned}$$

When $(M, w) \models \varphi$ for all $w \in M$, we write $M \models \varphi$. Similarly, we write $\models_{\mathcal{X}} \varphi$ to represent the fact that $M \models \varphi$ for all models M existing in a class \mathcal{X} .

Example 1 Let us suppose, that as far as we are concerned, there are only two relevant facts, namely, that Newton formulated the law of universal gravitation (p) and that Newton discovered infinitesimal calculus (q). Therefore, there exist only four possible descriptions of the world:

$$w_1 : p \wedge q \qquad w_2 : p \wedge \neg q \qquad w_3 : \neg p \wedge q \qquad w_4 : \neg p \wedge \neg q$$

Now let's suppose that agent i knows the first of these facts, but ignores the second. This means that w_1 and w_2 are, as far as agent i is concerned, possible descriptions of the real world that he cannot distinguish himself. Knowing that w_1 is the real world, we will consider that w_2 is accessible from w_1 for the agent i , or also that it is an *epistemic alternative* for agent i thanks to the accessibility relation (or indistinguishably relation) which is formalised as : $w_1 R_i w_2$. In the model sketched in Figure 1.1, the formulas $K_i p$, $\widehat{K}_i q$ and $\widehat{K}_i \neg q$ are true.

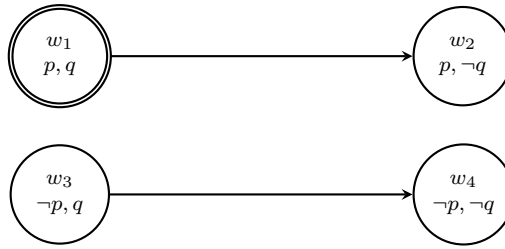


Figure 1.1: Possible worlds.

We shall start by presenting the possible axiomatisations of propositional epistemic logic (*PEL*) before proceeding to a critical discussion of their axioms. The simplest system, known as **K**, consists of the following axiom schemes (henceforth, for brevity, we will simply say axioms) and inference rules (for $\varphi, \psi \in PEL$, $i \in Atm$):

PC : Complete set of axioms for PC
 K : Distribution: $K_i\varphi \wedge K_i(\varphi \rightarrow \psi) \rightarrow K_i\psi$
 MP : Modus ponens: from φ and $\varphi \rightarrow \psi$ infer ψ
 NEC : Necessitation: from φ infer $K_i\varphi$

We can obtain stronger systems by adding some of the following axioms:

$$T : K_i\varphi \rightarrow \varphi \qquad 4 : K_i\varphi \rightarrow K_iK_i\varphi \qquad 5 : \neg K_i\varphi \rightarrow K_i\neg K_i\varphi$$

The systems derived from the addition of each of these axioms are known, for historical reasons, by the following names:

$$\mathbf{T} = \mathbf{K} + T \qquad \mathbf{S4} = \mathbf{T} + 4 \qquad \mathbf{S5} = \mathbf{S4} + 5$$

Standardly accepted properties of knowledge are that: known information is true (Axiom T), you are aware of - know - your knowledge (Axiom 4), and you are aware of your ignorance (Axiom 5). These axioms correspond to the structural properties of reflexivity, transitivity and Euclidicity. The complete list of axioms is shown in Table 1.1.


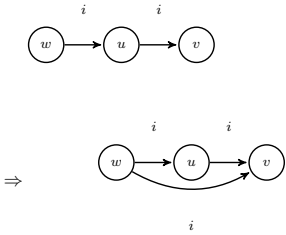
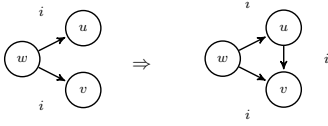
	Axiom	Condition on Kripke models	Diagram of the condition
K	$K_i\varphi \wedge K_i(\varphi \rightarrow \psi) \rightarrow K_i\psi$	<i>None</i>	
T	$K_i\varphi \rightarrow \varphi$	<i>Reflexivity</i> : for all $w, (w, w) \in R_i$	
4	$K_i\varphi \rightarrow K_iK_i\varphi$	<i>Transitivity</i> : for all $w, u, v, (w, u) \in R_i$ and $(u, v) \in R_i$ implies $(w, v) \in R_i$	
5	$\widehat{K}_i\varphi \rightarrow K_i\widehat{K}_i\varphi$	<i>Euclidean</i> : for all $w, u, v, (w, u) \in R_i$ and $(w, v) \in R_i$ implies $(u, v) \in R_i$	

Table 1.1: Structural properties of knowledge.

One critical aspect that appears in models based on EL is they are subjective to the logical omniscience problem (LOP): agents believe all logical consequences of what they believe. LOP is a major objection against EL because it makes unrealistic assumptions about the reasoning power of the agents. Two notions of belief have been proposed to tackle the omniscience problem: implicit and explicit belief. Implicit beliefs are closed under logical consequence, while explicit beliefs are not [Levesque 1984].

1.2 Doxastic logic

Doxastic logic is a type of logic concerned with reasoning about beliefs. However, as we said before, the term epistemic logic is often used generically to represent both knowledge and belief [Gabbay & Guenther 2003]. Regarding the basic axioms and rules used to specify doxastic logic at syntactic level, there are no essential modifications in comparison with those used by epistemic logic: all the tautologies of propositional logic and the axiom of distribution are still admitted as basic axioms, in addition to the rules of modus ponens and generalization of knowledge. However, axiom T seems to be too strong to fit in the definition of doxastic logic. With respect to knowledge it seems reasonable to postulate that everything someone knows is true, while in the case of a belief it is possible, or almost a general fact, that beliefs could be false. We use $B_i\varphi$ to represent the fact that agent i believes that φ . Thus, the truth axiom is manifestly excessive and we must settle for a weaker requirement. Therefore, in order to make the beliefs of our agents logically consistent, we must replace axiom T by the weaker one: $D = \neg(B_i\varphi \wedge B_i\neg\varphi)$, which states that for any φ that agent i accepts to believe, he can not believe its opposite $\neg\varphi$ at the same time. The basic system for propositional doxastic logic is represented by these axioms and is known as KD . We can add the axioms of positive (4) and negative (5) introspection to the basic system in order to generate the systems KD_4 and KD_45 respectively.

Regarding the semantics, a model M is defined above as a structure $M = \langle W, R, V \rangle$, where W is a non-empty set of possible worlds, R is a function mapping now a doxastic accessibility relation $R_i \subseteq W \times W$ for each agent $i \in Agt$ and $V : Atm \rightarrow 2^W$ is a valuation function that assigns to each $p \in Atm$ a truth value in each possible world $w \in W$.

The truth of a formula φ in a possible world w of a model $(M, w \models \varphi)$ is defined as above, substituting K for B and \widehat{K} for \widehat{B} . Thus, the difference between these operators lies in the properties required for axiom T , where the accessibility relation must be reflexive, while in axiom D , it is only required to be serial, i.e. for all $w \in M$ there is a $w' \in M$ such that wRw' .

1.3 Dynamic Epistemic Logic - DEL

DEL is the logic resulting from the combination of two operators. One for representing the agent's knowledge or belief and the other for representing the evolution of knowledge or belief as a consequence of the occurrence of an action (also called event).

Two of the most common types of actions are public and private announcements. Public Announcement Logic (PAL) incorporates an action which is used to inform all agents about a sentence and allows them to simultaneously accept this announcement. Private announcements are inside the group of more complex actions which involves passing information to an agent or a group of agents as a secret, hidden completely or partially from others. The latter actions of communication can be modeled

using *action models* [van Ditmarsch *et al.* 2007, van Benthem 2003]. Action models were first introduced under the name of *action structure* in [Baltag *et al.* 1998].

An *event model* is similar to a Kripke model, but instead of worlds it contains events with preconditions. Like epistemic models, event models describe indistinguishability for agents with a relation \sim_i for each of them.

Definition 6 (Action Model) *An action model is a tuple $\mathcal{E} = (E, (\sim_i)_{i \in \text{Agt}}, \text{pre})$ where:*

- E is a non-empty set of events,
- $\sim_i \subseteq E \times E$ is a binary accessibility relation for each agent $i \in \text{Agt}$,
- $\text{pre}: E \rightarrow \mathcal{L}_K(\text{Atm}, \text{Agt})$ assigns a precondition to each event.

The state-transition function of DEL is called *product update*. This operation allows applying an action model to an epistemic model to describe epistemic change. It generates a new epistemic model by performing a cross product between a Kripke model and an action model.

Definition 7 (Product Update) *Giving an epistemic model $M = \langle W, R, V \rangle$ and an action model $\mathcal{E} = (E, \sim_i, \text{pre})$, the product update is defined as $M \otimes \mathcal{E} = \langle W', R', V' \rangle$, where:*

- $W' = \{(w, e) \mid w \in W, e \in E \text{ and } (M, w) \models \text{pre}(e)\}$,
- $R'_i = \{((w, e), (w', e')) \in W' \times W' \mid wR_iw' \text{ and } e \sim_i e'\}$,
- $V'(p) = \{(w, e) \in W' \mid (M, w) \models p\}$.

Example 2 *Two friends, Alice and Bob, meet in a bar. Bob knows that Alice has submitted a paper for an important AI conference. He also knows that she is waiting for an email confirming whether the paper was accepted.*

Let's represent these two possible outcomes as $p =$ "the paper was accepted" and $\neg p =$ "the paper was not accepted". While they are talking, Alice checks her phone and exclaims: "I just got an email from the conference organizers". At this point, Alice reads the mail in front of Bob, but without telling him the content of it. Thus, Bob can not distinguish between Alice learning p or Alice learning $\neg p$. We are going to encapsulate these two actions in an action model and define these actions as e_1 and e_2 with preconditions p and $\neg p$ respectively, as shown in Figure 1.2.

Example 3 *In a variant of the previous example: At the moment that Alice indicates to Bob that she has just received an email from the conference organizers, Bob receives a call and has to leave the room to answer the call. Upon returning he sees that Alice is absent and that she has left her phone on the table.*

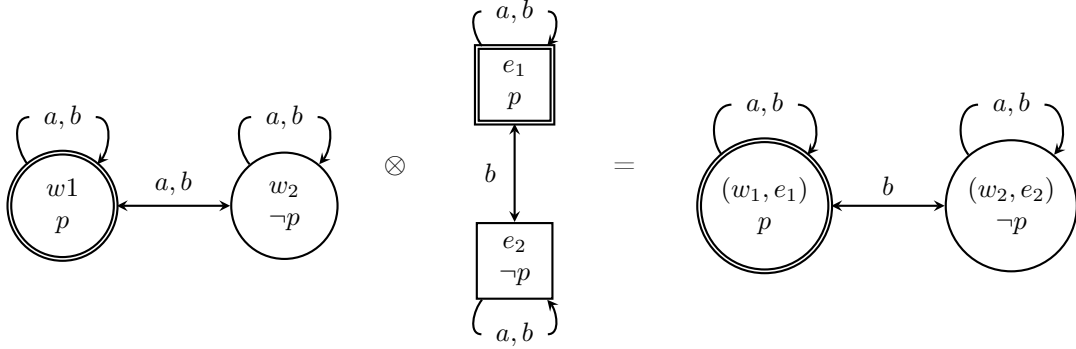


Figure 1.2: Dynamic operator effect on the model

Now Bob is not sure whether Alice has read the email or not. If the first scenario happened then Alice knows whether p . But if she hasn't read the email, then nothing changes and therefore the result is the same as before the email arrived. We represent this last scenario with \top in Figure 1.3.

Definition 8 (Language of dynamic epistemic logic) *The language of dynamic epistemic logic \mathcal{L}_{Del} extends \mathcal{L}_K by means of the addition of a dynamic operator:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [\mathcal{E}, e]\varphi,$$

where, K is the epistemic modality and $[\mathcal{E}, e]$ the dynamic modality (a pointed action model) with \mathcal{E} denoting an *event model* and $e \in \mathcal{E}$. We read $[\mathcal{E}, e]\varphi$ as “after the occurrence of (\mathcal{E}, e) , φ holds”.

Definition 9 (Satisfaction Relation) *We interpret the dynamic operator for action models as follows :*

$$(M, w) \models (\mathcal{E}, e) \iff M, w \models pre(e) \text{ implies } M \otimes \mathcal{E}, (w, e) \models \varphi.$$

Although EL and DEL allow the representation of two important epistemic aspects such as the deep-nested beliefs and the model dynamics by means of product update operations, these approaches present two main limitations. Firstly, in EL and DEL the difference between explicit and implicit beliefs can not be captured. Secondly, updating the epistemic model by means of private announcements in DEL implies that two copies of the model are maintained, one for the perceiver (the updated model) and the other for the non-perceivers (the original model), increasing the complexity of the satisfiability problem for DEL to NEXTPTIME-complete in a multi-agent environment.

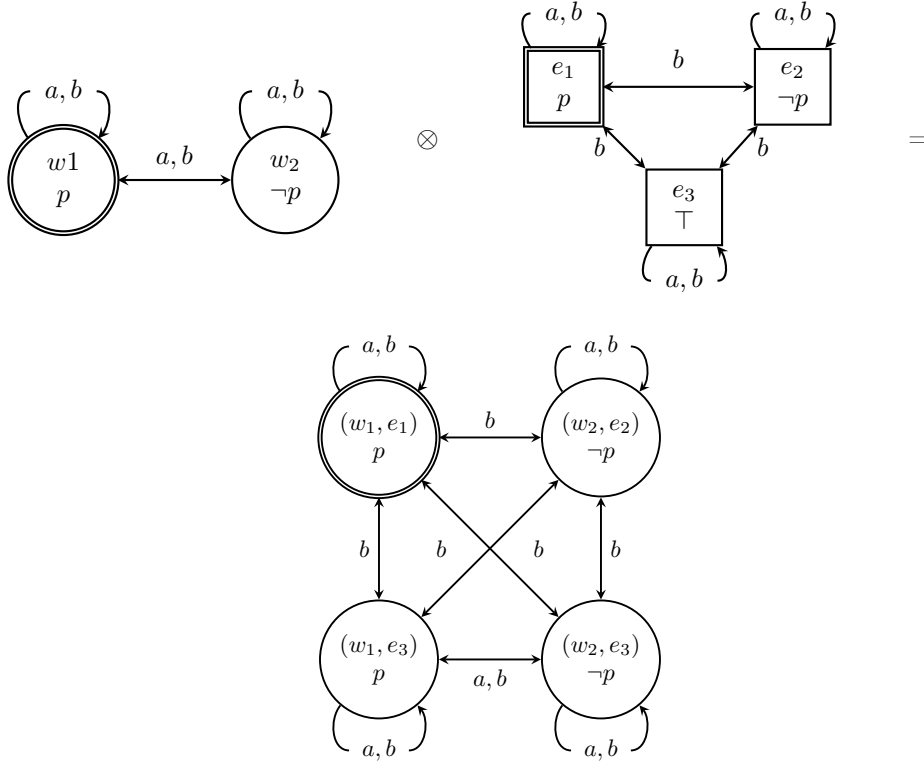


Figure 1.3: Dynamic operator effect on the model

1.4 Logic of Doxastic Attitudes - LDA

LDA is a multi-agent epistemic logic whose semantics exploits the concept of belief base [Lorini 2020, Lorini 2018]. In this logic it is possible to identify explicit beliefs, as a fact in an agent's belief base, from implicit beliefs, as those pieces of information that can be deducible from the agent's explicit beliefs. LDA provides a set of features that allow overcoming the EL and DEL limitations. First of all, a generalization of the standard EL approach is provided by LDA in order to capture the difference between explicit and implicit belief. Another advantage of LDA in comparison with DEL, is when we deal with information updates of type private belief expansion (PBE). In the former, the original epistemic model has to be duplicated by creating one copy of the model for the perceiver, in which their beliefs have changed and another copy for the non-perceivers, in which their beliefs have not changed leading to an exponential growth of the model in the length of the sequence of private announcements. In the latter the effects of the actions are represented only in the perceiver's belief base, while the belief base of the other agents remain unchanged. We are going to provide details about the syntax and the semantics of the logic LDA in Chapter 2.

1.5 Epistemic planning

According to [Ghallab *et al.* 2004] planning can be understood as the reasoning side of acting. Planning is the type of reasoning in which we try to establish what to do in order to make some arbitrary condition true. The condition that we want to achieve is called the goal, and the sequence of actions we seek that will make the goal true is called a plan [Brachman & Levesque 2004]. Actions to be considered can be of two types: ontic (physical) and epistemic actions (related to obtaining knowledge). Among the former we can mention the action of moving a package from one position to another, while among the latter we can consider observations and sensing actions [Herzig & de Lima 2006].

Epistemic planning extends classical planning by adding the epistemic concepts of knowledge and belief. Based on the fact that we are interested in modeling knowledge and belief in planning, it seems to be that DEL provides the conceptual framework over which we can specify our epistemic planning problem.

Definition 10 (Epistemic planning) *An epistemic planning problem is a triple $\langle \Sigma, Op, \alpha_G \rangle$, where Σ is a knowledge or a belief structure, Op is a finite set of operators and α_G is a formula from the epistemic language \mathcal{L}_K . We call the first component the initial situation, the second the available set of epistemic actions and the third the goal.*

A solution to an epistemic planning problem $\langle \Sigma, Op, \alpha_g \rangle$ is given by a finite sequence of actions $\epsilon_1, \dots, \epsilon_n$ from Op such that $\Sigma \models [\epsilon_1] \dots [\epsilon_n] \alpha_G$. The sequence of actions is what we call the *plan*. The size of the *plan* is the number of actions in the sequence.

Epistemic planning based on DEL allows us to represent the effect of the epistemic actions in the initial state by using action models.

Example 4 *In a variant of our Example 3, let us suppose now that Alice and Bod are co-authors of the paper and the goal is to make both agents knowing p without suspecting that each other does. Therefore, α_G is specified as follows: $K_a p \wedge K_b p \wedge \neg K_a K_b p \wedge \neg K_b K_a p$. Let ϵ_1 denote the epistemic action corresponding to a private announcement of p to Alice. Similarly, ϵ_2 is the epistemic action of informing Bod about the true of p . In terms of DEL, a solution to this epistemic planning task is the action sequence ϵ_1, ϵ_2 , since we have $\Sigma \otimes \epsilon_1 \otimes \epsilon_2 \models \alpha_G$.*

1.6 Satisfiability - SAT

Given a propositional formula φ , the satisfiability problem (SAT), consists of finding a set of values (either *true* or *false*) for each literal $\in \varphi$ in a way that φ evaluates to *true*. If this is the case, φ is called satisfiable. By contrast, if no such assignment exists, i.e., φ is *false* for all possible variable assignments, then the formula is unsatisfiable. For instance, the formula “ $p \wedge \neg q$ ” is satisfiable because it is possible to

assign the values of p to true and q to false, which makes $(p \wedge \neg q)$ true. Conversely, “ $p \wedge \neg p$ ” is unsatisfiable.

In addition, we know that only the set of atoms that are true are included in an interpretation. Therefore, the interpretation: $I = p, q$ assigns the value *true* to p and q , and *false* to all others. For example, the formula: $(p \wedge q \rightarrow r) \wedge (p \rightarrow q) \rightarrow (p \rightarrow r)$, would evaluate to false under I , because $I(p) = \text{true}$; $I(q) = \text{true}$; $I(r) = \text{false}$, so $I \models p \wedge q$ and $I \not\models p \wedge q \rightarrow r$.

Definition 11 (Validity and Satisfiability) *A formula φ is called valid iff it is true in all interpretations, i.e. $I \models \varphi$ for all interpretations I . We write $\models \varphi$ iff formula φ is valid. A formula φ is called satisfiable if there is an interpretation I in which φ is true, i.e. $I \models \varphi$. Otherwise it is called unsatisfiable.*

Satisfiability and validity are duals one another. As a result, a formula φ is valid if and only if $\neg\varphi$ is unsatisfiable. A proof of validity for φ from the unsatisfiability of $\neg\varphi$ is called a refutation.

SAT solvers are software tools which aim to solve the Boolean satisfiability problem. Most state-of-the-art SAT solvers are based on algorithms which include variants of Davis–Putnam–Logemann–Loveland (DPLL) [Davis *et al.* 1962] or conflict-driven clause learning (CDCL) [Marques Silva & Sakallah 1996] techniques. The main difference between the two approaches is that CDCL, unlike DPLL, is able to learn “from its mistakes”, i.e.: when it reaches a conflict, it backtracks to the conflict level and then applies the knowledge it has gathered from previous unsuccessful assignments to the *implication graph* it has generated.

Hence, the solver is able to ignore huge sections of the search space that will never satisfy the formula. MiniSAT [Sörensson & Een 2005] and Glucose [Audemard & Simon 2009] are among the SAT solvers that implement the CDCL algorithm.

Most of the current SAT solvers require the \mathcal{L}_{Prop} formula to be in its Conjunctive Normal Form - CNF.

Definition 12 (Conjunctive Normal Form (CNF)) *Let φ be a formula in its CNF. Then:*

- A literal l is a variable p or its negation $\neg p$,*
- A clause $C = (l_1 \vee \dots \vee l_m)$ is a disjunction over literals,*
- A formula φ is in CNF if it consists of a conjunction of clauses.*

In order to express a \mathcal{L}_{Prop} formula into its CNF form, the translation of it in its Negation Normal Form - NNF is required. A formula in NNF form may comprise of conjunctions, disjunctions, or literals. The negation symbol may only be found in literals. Every formula has an equivalent in NNF. Thus, the formula: $p \wedge (\neg q \vee r) \wedge \neg r) \vee s$ is in NNF form, while $\neg(p \wedge \neg q) \vee r$ is not.

SAT was the first NP-complete problem, and as such is among the most important problems in computer science. This problem has applications in inductive inference, database integrity, circuit synthesis and many others.

1.7 Quantified Boolean Formula - QBF

In computational complexity theory, the quantified Boolean formula problem (QBF) is a generalization SAT with the addition of existential and universal quantifiers that can be applied to each variable. To rephrase this, it asks whether a quantified sentential form over a set of Boolean variables is true or false. For example, the following is an instance of QBF: $\forall p \exists q, r. ((p \vee r) \wedge q)$. The standard encoding of a QBF formula is defined in PCNF (Prenex Conjunctive Normal Form): $\varphi ::= Q.\psi$, where:

$$\varphi ::= \underbrace{\forall p \exists q, r}_{\text{quantifier prefix } Q} \underbrace{((p \vee r) \wedge q)}_{\text{propositional CNF } \psi}$$

We require that no variable appears twice in Q and that all variables in ψ appear in Q (i.e., ψ contains no free variables).

Most current QBF solvers algorithms can be divided into classic DPLL based [Goultiaeva & Bacchus 2010, Klieber *et al.* 2010] and expansion based techniques [Bloem *et al.* 2018]. The former perform conflict and solution-driven search throughout the formula's assignment. In contrast, the latter eliminate quantifiers by expanding into boolean connectives. Experimental evaluation showed in [Bubeck & Büning 2007] demonstrates that expansion-based QBF solvers outperform DPLL-based solvers on a number of benchmark families. However, in the case of large size formulas, expansion-based solvers may lead to an exponential blowup. To mitigate this drawback [Janota *et al.* 2016] proposes an improving of the expansion-based technique using Counterexample Guided Abstraction Refinement (CEGAR) [Clarke *et al.* 2003]. Among the QBF-solvers that employ CEGAR are RaReQS [Janota *et al.* 2016] and GhostQ [Klieber *et al.* 2010].

With QBF, it is possible to compactly model problem-solving and planning in multiagent settings, thanks to its feature of combining universal and existential quantifiers.

In general it is the case that any NP problem can be compactly encoded in SAT while QBF allows us to compactly encode any PSPACE problem: QBF is PSPACE complete [Coste Marquis *et al.* 2006].

Logical Framework

This chapter is devoted to presenting the logic which will serve as a specification language for cognitive planning. We start by recalling the full language and the semantics presented in [Lorini 2018, Lorini 2020]. This language distinguishes explicit belief (a fact in an agent’s belief base) from implicit belief (a fact that is deducible from the agent’s explicit beliefs). Next, we study an NP-fragment of the logic, whose satisfiability problem is reduced to SAT. Finally, we focus on the extension of the latter by belief base expansion operators.

2.1 Full Language and Semantics

The language of our logic of explicit and implicit belief is defined in two steps. First, the language $\mathcal{L}_0(Atm, Agt)$ is defined by the following grammar in BNF:

$$\alpha ::= p \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \Delta_i\alpha,$$

where p ranges over Atm and i ranges over Agt . $\mathcal{L}_0(Atm, Agt)$ is the language for representing agents’ explicit beliefs. The formula $\Delta_i\alpha$ is read “ i explicitly believes that α ”. Then, the language $\mathcal{L}(Atm, Agt)$ extends the language $\mathcal{L}_0(Atm, Agt)$ by modal operators of implicit belief and is defined by the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_i\varphi \mid \Diamond_i\varphi,$$

where α ranges over $\mathcal{L}_0(Atm, Agt)$ and i ranges over Agt . For notational convenience we write \mathcal{L}_0 instead of $\mathcal{L}_0(Atm, Agt)$ and \mathcal{L} instead of $\mathcal{L}(Atm, Agt)$, when the context is unambiguous. The formula $\Box_i\varphi$ is read “ i implicitly believes that φ ” and $\Diamond_i\varphi$ is read “ φ is compatible (or consistent) with i ’s implicit beliefs”. The other Boolean constructions \top , \perp , \rightarrow and \leftrightarrow are defined in the standard way. We introduce here \vee and \Diamond_i as primitive and do not define them from \wedge and \Box_i because at a later stage we will need them for translating formulas in negation normal form.

The interpretation of language \mathcal{L} exploits the notion of belief base. While the notions of possible state (or world) and epistemic alternative are primitive in the standard Kripke semantics for epistemic logic, they are defined from the primitive concept of belief base in this semantics. In particular, a state is a composite object including a description of both the agents’ belief bases and the environment.¹

¹This is similar to the way states are modeled in the interpreted system semantics for multi-agent systems [Fagin *et al.* 1995, Lomuscio *et al.* 2017].

Definition 13 (State) A state is a tuple $B = (B_1, \dots, B_n, V)$ where: for every $i \in \text{Agt}$, $B_i \subseteq \mathcal{L}_0$ is agent i 's belief base; $V \subseteq \text{Atm}$ is the actual environment. The set of all states is noted \mathbf{S} .

Note that an agent's belief base B_i can be infinite. The sublanguage $\mathcal{L}_0(\text{Atm}, \text{Agt})$ is interpreted w.r.t. states, as follows:

Definition 14 (Satisfaction) Let $B = (B_1, \dots, B_n, V) \in \mathbf{S}$. Then:

$$\begin{aligned} B \models p &\iff p \in V, \\ B \models \neg\alpha &\iff B \not\models \alpha, \\ B \models \alpha_1 \wedge \alpha_2 &\iff B \models \alpha_1 \text{ and } B \models \alpha_2, \\ B \models \alpha_1 \vee \alpha_2 &\iff B \models \alpha_1 \text{ or } B \models \alpha_2, \\ B \models \Delta_i\alpha &\iff \alpha \in B_i. \end{aligned}$$

Observe in particular the set-theoretic interpretation of the explicit belief operator: agent i explicitly believes that α if and only if α is included in her belief base.

A multi-agent belief model (MAB) is defined to be a state supplemented with a set of states, called *context*. The latter includes all states that are compatible with the common ground [Stalnaker 2002], i.e., the body of information that the agents commonly believe to be the case.

Definition 15 (Multi-Agent Belief Model) A multi-agent belief model (MAB) is a pair (B, Cxt) , where $B \in \mathbf{S}$ and $\text{Cxt} \subseteq \mathbf{S}$. The class of all MABs is noted \mathbf{M} .

Note that we do not impose that $B \in \text{Cxt}$. When $\text{Cxt} = \mathbf{S}$ then (B, Cxt) is said to be *complete*, since \mathbf{S} is conceivable as the complete (or universal) context which contains all possible states. We compute an agent's set of epistemic alternatives from the agent's belief base, as follows.

Definition 16 (Epistemic alternatives) Let $i \in \text{Agt}$. Then \mathcal{R}_i is the binary relation on the set \mathbf{S} such that, for all $B = (B_1, \dots, B_n, V), B' = (B'_1, \dots, B'_n, V') \in \mathbf{S}$:

$$B\mathcal{R}_iB' \text{ if and only if } \forall\alpha \in B_i : B' \models \alpha.$$

$B\mathcal{R}_iB'$ means that B' is an epistemic alternative for agent i at B . So i 's set of epistemic alternatives at B includes exactly those states that satisfy all i 's explicit beliefs.

Definition 17 extends Definition 14 to the full language \mathcal{L} . Its formulas are interpreted with respect to MABs. We omit Boolean cases that are defined in the usual way.

$$\mathcal{L}_{\text{Frag}} \xrightarrow{nmf} \mathcal{L}_{\text{Frag}}^{\text{NNF}} \xrightarrow{tr_1} \mathcal{L}_{\text{Mod}} \xrightarrow{tr_2} \mathcal{L}_{\text{Prop}}$$

Figure 2.1: Summary of reduction process

Definition 17 (Satisfaction) *Let $(B, Cxt) \in \mathbf{M}$. Then:*

$$\begin{aligned} (B, Cxt) \models \alpha &\iff B \models \alpha, \\ (B, Cxt) \models \Box_i \varphi &\iff \forall B' \in Cxt, \\ &\quad \text{if } BR_i B' \text{ then } (B', Cxt) \models \varphi, \\ (B, Cxt) \models \Diamond_i \varphi &\iff \exists B' \in Cxt \text{ such that} \\ &\quad BR_i B' \text{ and } (B', Cxt) \models \varphi. \end{aligned}$$

A formula $\varphi \in \mathcal{L}$ is valid in the class \mathbf{M} , noted $\models_{\mathbf{M}} \varphi$, if and only if $(B, Cxt) \models \varphi$ for every $(B, Cxt) \in \mathbf{M}$; it is satisfiable in \mathbf{M} if and only if $\neg\varphi$ is not valid in \mathbf{M} .

Theorem 1 *Checking satisfiability of $\mathcal{L}(\text{Atm}, \text{Agt})$ formulas in the class \mathbf{M} is a PSPACE-hard problem.*

This theorem is a consequence of the fact that our logic contains the basic modal logic K whose satisfiability problem is PSPACE-complete [Halpern & Moses 1992].

2.2 NP-Complete Fragment

In this section, we study the following fragment of the language \mathcal{L} , called $\mathcal{L}_{\text{Frag}}$:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_{\mathbf{m}} \alpha \mid \Diamond_{\mathbf{m}} \alpha,$$

where α ranges over \mathcal{L}_0 and \mathbf{m} is a special agent in Agt called the ‘machine’. In $\mathcal{L}_{\text{Frag}}$, all agents have explicit beliefs but only agent \mathbf{m} has implicit beliefs, and moreover the latter are restricted to \mathcal{L}_0 formulas of type α . So there are no nested implicit beliefs for agent \mathbf{m} . Agent \mathbf{m} is assumed to be the unique artificial agent in the system which is endowed with unbounded reasoning and planning capabilities. The cognitive planning problem will be modeled from agent \mathbf{m} ’s perspective.

In the rest of this section, we are going to provide a polysize reduction of the satisfiability problem of $\mathcal{L}_{\text{Frag}}$ to SAT. The reduction consists of three steps which are summarized in Figure 2.1. As a first step, we put $\mathcal{L}_{\text{Frag}}$ formulas in negation

normal form (NNF) via the following function nnf :

$$\begin{aligned}
nnf(p) &= p, \\
nnf(\Delta_i \alpha) &= \Delta_i \alpha, \\
nnf(\Box_m \alpha) &= \Box_m nnf(\alpha), \\
nnf(\Diamond_m \alpha) &= \Diamond_m nnf(\alpha), \\
nnf(\varphi \wedge \psi) &= nnf(\varphi) \wedge nnf(\psi), \\
nnf(\varphi \vee \psi) &= nnf(\varphi) \vee nnf(\psi), \\
nnf(\neg p) &= \neg p, \\
nnf(\neg \Delta_i \alpha) &= \neg \Delta_i \alpha, \\
nnf(\neg \neg \varphi) &= nnf(\varphi), \\
nnf(\neg(\varphi \wedge \psi)) &= nnf(\neg \varphi \vee \neg \psi), \\
nnf(\neg(\varphi \vee \psi)) &= nnf(\neg \varphi \wedge \neg \psi), \\
nnf(\neg \Box_m \alpha) &= \Diamond_m nnf(\neg \alpha), \\
nnf(\neg \Diamond_m \alpha) &= \Box_m nnf(\neg \alpha).
\end{aligned}$$

Let us define the NNF variant $\mathcal{L}_0^{\text{NNF}}$ of the language \mathcal{L}_0 by the following grammar:

$$\beta ::= p \mid \neg p \mid \Delta_i \alpha \mid \neg \Delta_i \alpha \mid \beta_1 \wedge \beta_2 \mid \beta_1 \vee \beta_2,$$

where p ranges over Atm , i ranges over Agt and α ranges over \mathcal{L}_0 . Furthermore, let us define the language $\mathcal{L}_{\text{Frag}}^{\text{NNF}}$ by the following grammar. For β ranging over $\mathcal{L}_0^{\text{NNF}}$:

$$\varphi ::= \beta \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_m \beta \mid \Diamond_m \beta.$$

Proposition 1 *Let $\varphi \in \mathcal{L}_{\text{Frag}}$. Then, $\varphi \leftrightarrow nnf(\varphi)$ is valid in the class \mathbf{M} , and $nnf(\varphi) \in \mathcal{L}_{\text{Frag}}^{\text{NNF}}$.*

Note that the size of $nnf(\varphi)$ is polynomial in the size of φ .

As a second step, we define the following modal language \mathcal{L}_{Mod} into which the language $\mathcal{L}_{\text{Frag}}^{\text{NNF}}$ will be translated:

$$\begin{aligned}
\omega &::= q \mid \neg \omega \mid \omega_1 \wedge \omega_2 \mid \omega_1 \vee \omega_2, \\
\varphi &::= q \mid \neg \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \blacksquare \omega \mid \blacklozenge \omega,
\end{aligned}$$

where q ranges over the following set of atomic formulas:

$$Atm^+ = Atm \cup \{p_{\Delta_i \alpha} : i \in Agt \text{ and } \alpha \in \mathcal{L}_0(Atm, Agt)\}.$$

So $p_{\Delta_i \alpha}$ is nothing but a special propositional variable.

We interpret the language \mathcal{L}_{Mod} w.r.t. a pair (M, w) , called pointed Kripke model, where $M = (W, \Rightarrow, \pi)$, W is a non-empty set of worlds, $\Rightarrow \subseteq W \times W$ and $\pi : Atm^+ \longrightarrow 2^W$. (Boolean cases are again omitted as they are defined in the usual way.)

Definition 18 *The semantic interpretation for formulas in \mathcal{L}_{Mod} w.r.t. a pointed Kripke model (M, w) is as follows:*

$$\begin{aligned} (M, w) \models q &\iff w \in \pi(q); \\ (M, w) \models \blacksquare\omega &\iff \forall v \in W, \text{ if } w \Rightarrow v \text{ then } (M, v) \models \omega; \\ (M, w) \models \blacklozenge\omega &\iff \exists v \in W \text{ s.t. } w \Rightarrow v \text{ and } (M, v) \models \omega. \end{aligned}$$

The class of pointed Kripke models is noted \mathbf{K} . Satisfiability and validity of formulas in \mathcal{L}_{Mod} relative to the class \mathbf{K} is defined in the usual way.

Let $tr_1 : \mathcal{L}_{\text{Frag}}^{\text{NNF}} \rightarrow \mathcal{L}_{\text{Mod}}$ be a translation such that:

$$\begin{aligned} tr_1(p) &= p, \\ tr_1(\neg p) &= \neg p, \\ tr_1(\varphi_1 \wedge \varphi_2) &= tr_1(\varphi_1) \wedge tr_1(\varphi_2), \\ tr_1(\varphi_1 \vee \varphi_2) &= tr_1(\varphi_1) \vee tr_1(\varphi_2), \\ tr_1(\Delta_i \alpha) &= \begin{cases} p_{\Delta_m \alpha} \wedge \blacksquare tr_0(\alpha), & \text{if } i = m, \\ p_{\Delta_i \alpha}, & \text{otherwise,} \end{cases} \\ tr_1(\neg \Delta_i \alpha) &= \neg p_{\Delta_i \alpha}, \\ tr_1(\Box_m \beta) &= \blacksquare tr_0(\beta), \\ tr_1(\Diamond_m \beta) &= \blacklozenge tr_0(\beta); \end{aligned}$$

with $tr_0 : \mathcal{L}_0 \rightarrow \mathcal{L}_{\text{Mod}}$ such that:

$$\begin{aligned} tr_0(p) &= p, \\ tr_0(\neg \alpha) &= \neg tr_0(\alpha), \\ tr_0(\alpha_1 \wedge \alpha_2) &= tr_0(\alpha_1) \wedge tr_0(\alpha_2), \\ tr_0(\alpha_1 \vee \alpha_2) &= tr_0(\alpha_1) \vee tr_0(\alpha_2), \\ tr_0(\Delta_i \alpha) &= p_{\Delta_i \alpha}. \end{aligned}$$

As the following theorem indicates, the polynomial translation tr_1 guarantees the transfer of satisfiability from model class \mathbf{M} to model class \mathbf{K} .

Theorem 2 *Let $\varphi \in \mathcal{L}_{\text{Frag}}^{\text{NNF}}$. Then, φ is satisfiable in the class \mathbf{M} if and only if $tr_1(\varphi)$ is satisfiable in the class \mathbf{K} .*

SKETCH OF PROOF. The proof relies on the fact that the belief base semantics for the language $\mathcal{L}_{\text{Frag}}$ is equivalent to a “weaker” semantics exploiting pointed structures of the form (X, s) where $X = (S, \mathcal{B}, (\Rightarrow_i)_{i \in \text{Agt}}, \tau)$, S is a non-empty set of states, $s \in S$ is the actual state, $\mathcal{B} : \text{Agt} \times S \rightarrow 2^{\mathcal{L}_0}$ is a belief base function, $\tau : \text{Atm} \rightarrow 2^S$ is valuation function, $\Rightarrow_i \subseteq S \times S$ is agent i 's epistemic accessibility relation and with respect to which \mathcal{L} -formulas are interpreted as follows (boolean cases are omitted for simplicity): (i) $(X, s) \models p$ iff $s \in \tau(p)$, (ii) $(X, s) \models \Delta_i \alpha$ iff $\alpha \in \mathcal{B}(i, s)$, (iii)

$(X, s) \models \Box_i \varphi$ iff $\forall s' \in S$, if $s \Rightarrow_i s'$ then $(X, s') \models \varphi$. In particular, for every $\varphi \in \mathcal{L}_{\text{Frag}}$, we have that φ is satisfiable in \mathbf{M} iff φ is satisfiable in the subclass of pointed structures (X, s) such that $\Rightarrow_m(s) \subseteq \bigcap_{\alpha \in \mathcal{B}(m, s)} \|\alpha\|_{(X, s)}$ with $\|\alpha\|_{(X, s)} = \{s' \in X : (X, s') \models \alpha\}$. ■

As a last step, we provide a polysize reduction of \mathcal{L}_{Mod} -satisfiability to SAT, where the underlying propositional logic language $\mathcal{L}_{\text{Prop}}$ is built from the following set of atomic propositions:

$$\begin{aligned} \text{Atm}^{++} = & \{q_x : q \in \text{Atm}^+ \text{ and } x \in \mathbb{N}\} \cup \\ & \{r_{x,y} : x, y \in \mathbb{N}\}. \end{aligned}$$

The set Atm^{++} includes two types of atomic propositions: one of the form q_x denoting the fact that q is true at world x and the other of the form $r_{x,y}$ denoting the fact that world x is related to world y .

Let $tr_2 : \mathcal{L}_{\text{Mod}} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathcal{L}_{\text{Prop}}$ be the following translation function:

$$\begin{aligned} tr_2(q, x, y) &= q_x, \\ tr_2(\neg\varphi, x, y) &= \neg tr_2(\varphi, x, y), \\ tr_2(\varphi_1 \wedge \varphi_2, x, y) &= tr_2(\varphi_1, x, y) \wedge tr_2(\varphi_2, x, y), \\ tr_2(\varphi_1 \vee \varphi_2, x, y) &= tr_2(\varphi_1, x, y) \vee tr_2(\varphi_2, x, y), \\ tr_2(\blacksquare\omega, x, y) &= \bigwedge_{0 \leq z \leq y} (r_{x,z} \rightarrow tr_2(\omega, z, y)), \\ tr_2(\blacklozenge\omega, x, y) &= \bigvee_{0 \leq z \leq y} (r_{x,z} \wedge tr_2(\omega, z, y)). \end{aligned}$$

Translation tr_2 is similar to the translation of modal logic S5 into propositional logic given in [Caridroit *et al.* 2017] and, more generally, to the standard translation of modal logic into FOL in which accessibility relations are encoded by special predicates. The size of an \mathcal{L}_{Mod} formula, $size(\varphi)$, is defined by:

$$\begin{aligned} size(p) &= 1, \\ size(\varphi_1 \wedge \varphi_2) &= size(\varphi_1) + size(\varphi_2) + 1, \\ size(\varphi_1 \vee \varphi_2) &= size(\varphi_1) + size(\varphi_2) + 1, \\ size(\neg\varphi) &= size(\varphi) + 1, \\ size(\blacksquare\omega) &= size(\blacklozenge\omega) = size(\omega) + 1. \end{aligned}$$

Note that the size of $tr_2(\varphi, 0, size(\varphi))$ is polynomial in the size of φ .

Theorem 3 *Let $\varphi \in \mathcal{L}_{\text{Mod}}$. Then, φ is satisfiable in the class \mathbf{K} if and only if $tr_2(\varphi, 0, size(\varphi))$ is satisfiable in propositional logic.*

SKETCH OF PROOF. The theorem is proved in the same way as the standard translation of modal logic to FOL plus a straightforward adaptation of [Ladner 1977,

Lemma 6.1] about polysize-model property for S5 to our case. ■

The size of $tr_2(\varphi, 0, size(\varphi))$ being polynomial in the size of φ , thanks to Proposition 1, Theorem 2 and Theorem 3 we state the following complexity result.

Theorem 4 *Checking satisfiability of formulas in $\mathcal{L}_{\text{Frag}}$ in the class \mathbf{M} is an NP-complete problem.*

2.3 Dynamic Extension

In this section, we extend the language $\mathcal{L}_{\text{Frag}}$ by belief expansion operations. Specifically, we introduce the following language $\mathcal{L}_{\text{Frag}}^+$:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_m \alpha \mid \Diamond_m \alpha \mid [+_i\alpha]\varphi,$$

where α ranges over \mathcal{L}_0 and i ranges over Agt . The formula $[+_i\alpha]\varphi$ is read “ φ holds after agent i has privately expanded her belief base with α ”. Events of type $+_i\alpha$ are generically called informative actions.

Our extension has the following semantics relative to a MAB:

Definition 19 (Satisfaction relation, cont.) *Let $B = (B_1, \dots, B_n, V) \in \mathbf{S}$ and let $(B, Cxt) \in \mathbf{M}$. Then:*

$$(B, Cxt) \models [+_i\alpha]\varphi \iff (B^{+_i\alpha}, Cxt) \models \varphi$$

with $V^{+_i\alpha} = V$, $B_i^{+_i\alpha} = B_i \cup \{\alpha\}$ and $B_j^{+_i\alpha} = B_j$ for all $j \neq i$.

Intuitively speaking, the private expansion of i 's belief base by α simply consists of agent i adding the information that α to her belief base, while all other agents keep their belief bases unchanged.

The following equivalences are valid in the class \mathbf{M} :

$$\begin{aligned} [+_i\alpha]\alpha' &\leftrightarrow \begin{cases} \top, & \text{if } \alpha' = \Delta_i\alpha, \\ \alpha', & \text{otherwise;} \end{cases} \\ [+_i\alpha]\neg\varphi &\leftrightarrow \neg[+_i\alpha]\varphi; \\ [+_i\alpha](\varphi_1 \wedge \varphi_2) &\leftrightarrow [+_i\alpha]\varphi_1 \wedge [+_i\alpha]\varphi_2; \\ [+_i\alpha](\varphi_1 \vee \varphi_2) &\leftrightarrow [+_i\alpha]\varphi_1 \vee [+_i\alpha]\varphi_2; \\ [+_i\alpha]\Box_m \alpha' &\leftrightarrow \begin{cases} \Box_m(\alpha \rightarrow \alpha'), & \text{if } i = m, \\ \Box_m \alpha', & \text{otherwise;} \end{cases} \\ [+_i\alpha]\Diamond_m \alpha' &\leftrightarrow \begin{cases} \Diamond_m(\alpha \wedge \alpha'), & \text{if } i = m, \\ \Diamond_m \alpha', & \text{otherwise.} \end{cases} \end{aligned}$$

Thanks to these equivalences we can define the following reduction red transforming every $\mathcal{L}_{\text{Frag}}^+$ formula φ into an equivalent $\mathcal{L}_{\text{Frag}}$ formula $red(\varphi)$:

$$\begin{aligned}
red(p) &= p, \\
red(\Delta_i \alpha) &= \Delta_i \alpha, \\
red(\neg \varphi) &= \neg red(\varphi), \\
red(\varphi_1 \wedge \varphi_2) &= red(\varphi_1) \wedge red(\varphi_2), \\
red(\varphi_1 \vee \varphi_2) &= red(\varphi_1) \vee red(\varphi_2), \\
red(\Box_m \varphi) &= \Box_m red(\varphi), \\
red(\Diamond_m \varphi) &= \Diamond_m red(\varphi), \\
red([+_i \alpha] \alpha') &= \begin{cases} \top, & \text{if } \alpha' = \Delta_i \alpha, \\ red(\alpha'), & \text{otherwise;} \end{cases} \\
red([+_i \alpha] \neg \varphi) &= red(\neg [+_i \alpha] \varphi), \\
red([+_i \alpha](\varphi_1 \wedge \varphi_2)) &= red([+_i \alpha] \varphi_1 \wedge [+_i \alpha] \varphi_2), \\
red([+_i \alpha](\varphi_1 \vee \varphi_2)) &= red([+_i \alpha] \varphi_1 \vee [+_i \alpha] \varphi_2), \\
red([+_i \alpha] \Box_m \alpha') &= \begin{cases} red(\Box_m(\alpha \rightarrow \alpha')), & \text{if } i = m, \\ red(\Box_m \alpha') & \text{otherwise;} \end{cases} \\
red([+_i \alpha] \Diamond_m \alpha') &= \begin{cases} red(\Diamond_m(\alpha \wedge \alpha')) & \text{if } i = m, \\ red(\Diamond_m \alpha') & \text{otherwise;} \end{cases} \\
red([+_i \alpha_1][+_j \alpha_2] \varphi) &= red([+_i \alpha_1] red([+_j \alpha_2] \varphi)).
\end{aligned}$$

Proposition 2 *Let $\varphi \in \mathcal{L}_{\text{Frag}}^+$. Then, $\varphi \leftrightarrow red(\varphi)$ is valid in the class \mathbf{M} , and $red(\varphi) \in \mathcal{L}_{\text{Frag}}$.*

The following theorem is a consequence of Theorem 4, Proposition 2 and the fact that the size of $red(\varphi)$ is polynomial in the size of φ .

Theorem 5 *Checking satisfiability of formulas in $\mathcal{L}_{\text{Frag}}^+$ in the class \mathbf{M} is an NP-complete problem.*

Before concluding this section, we define the concept of logical consequence for the language $\mathcal{L}_{\text{Frag}}^+$ which will be used in the formulation of the cognitive planning problem in Chapter 3. Let Σ be a finite subset of \mathcal{L}_0 and let $\varphi \in \mathcal{L}_{\text{Frag}}^+$. We say that φ is a logical consequence of Σ in the class \mathbf{M} , noted $\Sigma \models_{\mathbf{M}} \varphi$, if and only if, for every $(B, Cxt) \in \mathbf{M}$ such that $Cxt \subseteq \mathbf{S}(\Sigma)$ we have $(B, Cxt) \models \varphi$, with $\mathbf{S}(\Sigma) = \{B \in \mathbf{S} : \forall \alpha \in \Sigma, B \models \alpha\}$. We say that φ is Σ -satisfiable in the class \mathbf{M} if and only if, $\neg \varphi$ is not a logical consequence of Σ in \mathbf{M} . Clearly, φ is valid if and only if φ is a logical consequence of \emptyset , and φ is satisfiable if and only if φ is \emptyset -satisfiable.

As the following deduction theorem indicates, the logical consequence problem with a finite set of premises can be reduced to the satisfiability problem.

Theorem 6 *Let $\varphi \in \mathcal{L}_{\text{Frag}}^+$ and let $\Sigma \subset \mathcal{L}_0$ be finite. Then, $\Sigma \models_{\mathbf{M}} \varphi$ if and only if $\models_{\mathbf{M}} \bigwedge_{\alpha \in \Sigma} \Box_m \alpha \rightarrow \varphi$.*

2.4 Conclusion

This chapter started from the multi-agent epistemic logic presented in [Lorini 2020]. The logic distinguishes between explicit and implicit belief and is interpreted relative to a semantic using belief bases.

Given that the satisfiability checking problem for this logic is PSPACE complete, we studied a fragment that considers a mono-modal version of the language with a single reasoning agent. We proposed a sequence of translations to transform formulas expressed in our fragment into propositional logic. We demonstrated that checking satisfiability of formulas in our fragment is an NP-complete problem.

Finally, we extended our NP-fragment by dynamic operators of belief expansion. We found that adding this extension does not increase the complexity for checking satisfiability of formulas in our NP-fragment.

Cognitive Planning

In Chapter 2, we studied an NP-complete logic of explicit and implicit belief. We reduce its satisfiability checking problem to SAT. This chapter presents a novel approach to cognitive planning based on this logic, i.e., a planning agent aimed at changing another agent’s cognitive attitudes, including her beliefs and intentions. We propose a general architecture that considers the modules and data structures needed to perform the cognitive planning task. Afterwards, we formalize two cognitive planning problems, informative planning and interrogative planning. We encode both types of planning problems in our NP-fragment.

Moreover, we illustrate the potential for application of our model by means of two examples. In the first example, an artificial agent interacts with a human agent through dialogue and tries to induce in her a potential intention to practice a sport. In the second example, the artificial agent acts as a counselor, applying the principles of motivational interviewing [Lundahl & Burke 2009] in order to make the human aware of the inconsistency between her desires and her current behavior. Next, the artificial counselor tries to promote a positive behavior change in the person, by performing a sequence of assertions oriented to convince the human about the necessity of changing her behavior in order to achieve her goals.

Finally, this chapter presents an optimal encoding for the informative planning problem using QBF. Furthermore, we provide complexity results for finding a solution plan for the informative planning problem.

3.1 Related Work on Epistemic Planning and Persuasion

In social sciences, influence is defined as “change in an individual’s thoughts, feelings, attitudes, or behaviors that results from interaction with another individual or a group” [Rashotte 2009]. It is conceived as tightly connected with persuasion. The latter is the intentional form of influence in which an agent (the persuader) tries to make someone (the persuadee) do or believe something by giving her a good reason [Cialdini 2001, Perloff 2003].

Models of persuasion in AI are mostly based on argumentation. (See [Prakken 2006] for a general introduction to the research in this area.) Some of these models are built on Walton & Krabbe’s notion of persuasion dialogue in which one

party seeks to persuade another party to adopt a belief or point-of-view she does not currently hold [Walton & Krabbe 1995]. There exist models based on abstract argumentation [Bench-Capon 2003, Bonzon & Maudet 2011, Amgoud *et al.* 2000] as well probabilistic models where the persuader’s uncertainty about what the persuadee knows or believes is represented [Hunter 2015]. There exist also models based on possibility theory in which a piece of information is represented as an argument which can be more or less accepted depending on the trustworthiness of the agent who proposes it [Da Costa Pereira *et al.* 2011]. Persuasion has also been formalized with the support of logical tools, e.g., by combining abstract argumentation with dynamic epistemic logic (DEL) [Proietti & Yuste-Ginel 2019] and epistemic logic with dynamic logic [Budzyńska & Kacprzak 2008].

Epistemic planning is a generalization of classical planning that has been increasingly studied in AI in the last years. The goal to be achieved is not necessarily a state of the world but some belief states of one or more agents. This requires a theory of mind by the planning agent [Goldman 2006]. A typical goal in epistemic planning is to make a certain agent believe something. Such a belief of the persuadee may be a higher-order belief, i.e., a belief about another agents’ beliefs. The initial proposal was to use a standard logic of knowledge or belief together with a representation of actions in terms of event models of DEL [Bolander & Andersen 2011, Löwe *et al.* 2011]. While the DEL framework is very expressive, it turned out that the existence of a solution becomes quickly undecidable even for very simple kinds of event models [Aucher & Bolander 2013, Bolander *et al.* 2015a, Lê Cong *et al.* 2018]. Kominis and Geffner considered epistemic planning problems with very simple event models leading to a decidable fragment [Kominis & Geffner 2015]. They distinguish three kinds of actions: physical actions modifying the world, public updates (DEL-like public announcements), and sensing actions by means of which an agent learns whether a formula is true. Other researchers investigated another source of complexity, namely that of standard epistemic logic. There, reasoning is strictly more complex than in classical logic: the satisfiability problem is at least in PSPACE [Halpern & Moses 1992]. Based on earlier work by Levesque, Muise *et al.* studied epistemic planning in fragments of standard epistemic logic [Muise *et al.* 2015, Muise *et al.* 2021]. They considered state descriptions in terms of conjunctions of epistemic literals: formulas that do not contain any conjunction or disjunction. Cooper *et al.* considered another fragment: boolean combinations of ‘knowing-whether’ operators followed by propositional variables [Cooper *et al.* 2016].

Our approach pushes the envelope of the above approaches to epistemic planning. Our main contribution is its generalization to cognitive planning: it is not only some belief state of a target agent that is to be achieved, but more generally a cognitive state. The latter could involve not only beliefs, but also intentions. Cognitive planning makes clear the distinction between *persuasion on beliefs* (i.e., inducing someone to believe that a certain fact is true) and *persuasion on intentions* (i.e., inducing someone to form a certain intention) and elucidates the connection between these two notions. Specifically, since beliefs are the input of decision-making and

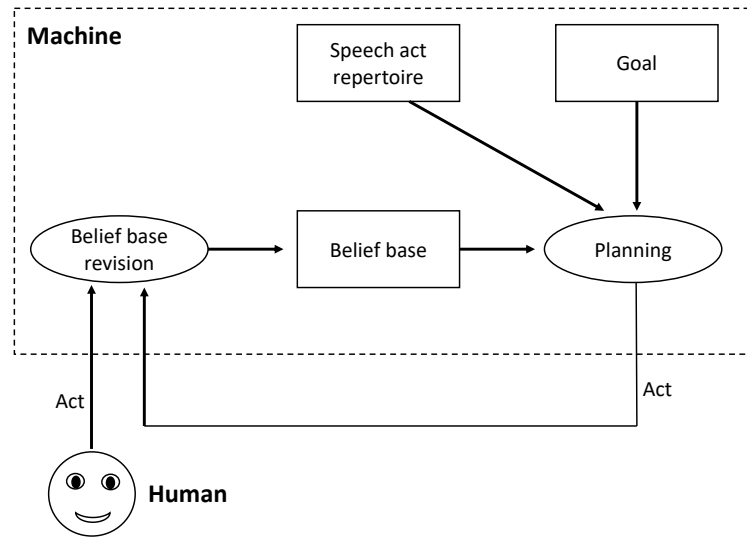


Figure 3.1: General architecture

provide reasons for deciding and for acting, the persuader can indirectly change the persuadee’s intentions by changing her beliefs, through the execution of a sequence of speech acts. In other words, in cognitive planning, the persuader tries to modify the persuadee’s beliefs *in order to* affect persuadee’s intentions. Moreover, cognitive planning takes into consideration resource boundedness and limited rationality of the interlocutor agent. This makes cognitive planning a very well-suited model for implementing motivational interviewing in human-machine interaction (HMI) applications in which an artificial agent is expected to interact with a human — who is by definition resource-bounded — through dialogue and to induce her to behave in a certain way.

3.2 General Architecture

The general architecture of our system is detailed in Figure 3.1.

Data structures The artificial planning agent, that for simplicity we call the machine, is endowed with three kinds of data structure: its belief base, the goal to be achieved and the repertoire of speech acts (or communicative actions) it can perform. We assume the machine’s action repertoire includes two types of speech act: assertions and questions. The machine can have persuading goals, aimed at changing the human’s beliefs, or influencing goals, aimed at inducing the human to form a certain intention or to behave in a certain way. The machine’s belief base includes both information about the environment and information about the human’s overall cognitive state and its way of functioning. In other words, the machine has a theory of the human’s mind. The machine’s belief base evolves during its dialogue with the human.

Interrogative and informative phase The interaction between the machine and the human is structured in two phases the *interrogative* (or *exploratory*) phase and the *informative* phase. In the interrogative phase the machine gathers information about the human's cognitive state. This includes information about the human's beliefs, desires and preferences. The interrogative phase is identified with a sequence of questions by the machine to the human. The informative phase is the core of the influence process. In this phase, the machine performs a sequence of assertions aimed at modifying the human's cognitive state (her beliefs and/or intentions). The interrogative phase is propaedeutic to the informative phase. Indeed, for the machine to be able to lead the human to change her behavior, it must have information about the human's cognitive state. Such an information is acquired during the interrogative phase. In this work, we assume that the two phases are unified at the planning level: the machine includes in its plan not only the assertions but also the questions. In particular, the machine has to find a sequence of questions followed by a sequence of assertions such that, for some possible answer by the human, the composition of the two sequences guarantees that the persuading or influencing goal will be achieved. It is reasonable to assume that the machine first tries to find a plan with only assertions. (why asking questions to the human if what the machine knows about the human's cognitive state is already sufficient to persuade or influence her). However, in most cases, the machine has uncertainty and lacks information about the human's cognitive state so that it must ask questions to the human before trying to induce her attitude change.

Execution of the plan After having selected a plan, the machine executes it. The machine can either execute the entire plan or execute it one piece after the other by waiting the reply of the human before executing the next piece. We assume that how the plan is executed depends on the application under consideration and on the type of speech act in the plan to be executed. It is reasonable to suppose that when executing the interrogative part of the plan, the machine asks a single question at each step and waits the answer by the human before moving to the next question. After each question by the machine, the human gives an answer and the machine expands or revises, when necessary, its belief base accordingly. Indeed, the information provided by the human in response to the machine's question can enrich the machine's belief base with new facts about the environment (objective facts) or about the human's cognitive state (mental facts) or make the machine's belief base inconsistent. In the latter case, the machine must revise its belief base after having incorporated the new information.

3.3 Planning Problems

In this section, we specify the cognitive planning problem in a two-agent version of the language $\mathcal{L}_{\text{Frag}}^+$ presented in chapter 2. Thus, we consider this time a finite set of agents $\text{Agt} = \{\mathfrak{h}, \mathfrak{m}\}$, with \mathfrak{h} denoting the human and \mathfrak{m} the machine.

The cognitive planning problem consists of finding a sequence of questions or informative actions for agent \mathbf{m} which guarantees that it believes that its goal α_G is satisfied. Agent \mathbf{m} is assumed to be an artificial agent which interacts with the resource-bounded human agent \mathbf{h} .

Informative actions Let $Act_{\mathbf{m}} = \{+_{\mathbf{m}}\alpha : \alpha \in \mathcal{L}_0\}$ be agent \mathbf{m} 's set of belief expansion operations (or informative actions) and let elements of $Act_{\mathbf{m}}$ be noted $\epsilon, \epsilon', \dots$. Speech acts of type ‘assertion’ are formalized as follows:

$$assert(\mathbf{m}, \mathbf{h}, \alpha) \stackrel{\text{def}}{=} +_{\mathbf{m}} \Delta_{\mathbf{h}} \Delta_{\mathbf{m}} \alpha.$$

The event $assert(\mathbf{m}, \mathbf{h}, \alpha)$ captures the speech act “agent \mathbf{m} asserts to agent \mathbf{h} that α ”. The latter is assumed to coincide with the perlocutionary effect [Searle 1969, Sect. 6.2] of the speaker learning that the hearer has learnt that the speaker believes that α .¹ We distinguish simple assertions from convincing actions:

$$convince(\mathbf{m}, \mathbf{h}, \alpha) \stackrel{\text{def}}{=} +_{\mathbf{m}} \Delta_{\mathbf{h}} \alpha.$$

The event $convince(\mathbf{m}, \mathbf{h}, \alpha)$ captures the action “agent \mathbf{m} convinces agent \mathbf{h} that α ”. We have $assert(\mathbf{m}, \mathbf{h}, \alpha) = convince(\mathbf{m}, \mathbf{h}, \Delta_{\mathbf{m}} \alpha)$. We assume ‘to assert’ and ‘to convince’ correspond to different utterances. While ‘to assert’ corresponds to the speaker’s utterances of the form “I think that α is true!” and “In my opinion, α is true!”, ‘to convince’ corresponds to the speaker’s utterances of the form “ α is true!” and “it is the case that α !”.

The previous abbreviations and, more generally, the idea of describing speech acts of a communicative plan performed by agent \mathbf{m} with \mathbf{m} 's private belief expansion operations is justified by the fact that we model cognitive planning from the perspective of the planning agent \mathbf{m} . Therefore, we only need to represent the effects of actions on agent \mathbf{m} 's beliefs.

Questions We consider binary questions by the machine \mathbf{m} to the human \mathbf{h} of the form $?_{\mathbf{m}, \mathbf{h}} \alpha$.² The set of binary questions is noted $Que_{\mathbf{m}}$. Intuitively, $?_{\mathbf{m}, \mathbf{h}} \alpha$ is the utterance performed by agent \mathbf{m} and directed to agent \mathbf{h} of the form “Do you think that α is true?”. Let elements of $Que_{\mathbf{m}}$ be noted λ, λ', \dots . Each question is associated with its set of possible answers. The answer function $\mathcal{A} : Que_{\mathbf{m}} \rightarrow 2^{Act_{\mathbf{m}}}$ is used to map each binary question to its set of possible answers and is defined as follows:

$$\mathcal{A}(?_{\mathbf{m}, \mathbf{h}} \alpha) = \{ +_{\mathbf{m}} \Delta_{\mathbf{h}} \alpha, +_{\mathbf{m}} \neg \Delta_{\mathbf{h}} \alpha \}.$$

Answers to binary questions are noted ρ, ρ', \dots . The operation $+_{\mathbf{m}} \Delta_{\mathbf{h}} \alpha$ captures agent \mathbf{h} 's positive answer to agent \mathbf{m} 's binary question $?_{\mathbf{m}, \mathbf{h}} \alpha$ (“I think that α is

¹We implicitly assume that, by default, \mathbf{m} believes that \mathbf{h} trusts its sincerity, so that \mathbf{h} will believe that \mathbf{m} believes what it says.

²In speech act theory, binary (yes-no) questions are usually distinguished from open questions.

true!”), while $+_{\mathbf{m}}\neg\Delta_{\mathbf{h}}\alpha$ captures agent \mathbf{h} 's negative answer (“I don't think that α is true!”). Note that if agent \mathbf{h} answers negatively to the consecutive questions $?_{\mathbf{m},\mathbf{h}}\alpha$ and $?_{\mathbf{m},\mathbf{h}}\neg\alpha$, then she expresses her uncertainty about the truth value of α .

We assume that the positive answer is the *default* answer to a question. Indeed, when agent \mathbf{m} asks question $?_{\mathbf{m},\mathbf{h}}\alpha$, it wants to verify whether agent \mathbf{h} endorses the belief that α and presupposes that agent \mathbf{h} will answer positively to the question. In this perspective, the speaker expects a confirmation by the interlocutor. Thus, for notational convenience, we write $da(?_{\mathbf{m},\mathbf{h}}\alpha)$ to denote the default answer $+_{\mathbf{m}}\Delta_{\mathbf{h}}\alpha$ to the question $?_{\mathbf{m},\mathbf{h}}\alpha$.

The following abbreviation defines a dynamic operator capturing the necessary effects of agent \mathbf{m} 's question:

$$[\lambda]\varphi \stackrel{\text{def}}{=} \bigwedge_{\rho \in \mathcal{A}(\lambda)} [\rho]\varphi,$$

with $\lambda \in \text{Que}_{\mathbf{m}}$. Note that, unlike the basic belief expansion operator $[+_{\mathbf{m}}\alpha]$, the operator $[\lambda]$ is non-deterministic, as it represents the consequences of *all possible answers* to question λ . In fact, while the formula $[+_{\mathbf{m}}\alpha]\neg\varphi \vee [+_{\mathbf{m}}\alpha]\varphi$ is valid in the class \mathbf{M} , the formula $[\lambda]\neg\varphi \vee [\lambda]\varphi$ is not.

Executability preconditions The set of events includes both informative actions and questions, and is defined as follows: $\text{Evt}_{\mathbf{m}} = \text{Act}_{\mathbf{m}} \cup \text{Que}_{\mathbf{m}}$. Elements of $\text{Evt}_{\mathbf{m}}$ are noted γ, γ', \dots . They have executability preconditions that are specified by the following function: $\mathcal{P} : \text{Evt}_{\mathbf{m}} \rightarrow \mathcal{L}_{\text{Frag}}$. We assume that an event γ can take place if its executability precondition $\mathcal{P}(\gamma)$ holds.

We use the executability precondition function \mathcal{P} to define the following operator of possible occurrence of an event:

$$\langle\langle\gamma\rangle\rangle\varphi \stackrel{\text{def}}{=} \mathcal{P}(\gamma) \wedge [\gamma]\varphi,$$

with $\gamma \in \text{Evt}$. The abbreviation $\langle\langle\gamma\rangle\rangle\varphi$ has to be read “the event γ can take place and φ necessarily holds after its occurrence”.

Informative and interrogative planning problems We conclude this section with a formal specification of two planning problems, informative planning and interrogative planning.

Definition 20 (Informative planning problem) An *informative planning problem* is a tuple $\langle \Sigma, \text{Op}_{\text{inf}}, \alpha_G \rangle$ where:

- $\Sigma \subset \mathcal{L}_0$ is a finite set of agent \mathbf{m} 's available information,
- $\text{Op}_{\text{inf}} \subset \text{Act}_{\mathbf{m}}$ is a finite set of agent \mathbf{m} 's informative actions,
- $\alpha_G \in \mathcal{L}_0$ is agent \mathbf{m} 's goal.

Informally speaking, an informative planning problem is the problem of finding an executable sequence of informative actions which guarantees that, at the end of the sequence, the planning agent \mathbf{m} believes that its goal α_G is achieved. Typically, α_G is a persuading or influencing goal, i.e., the goal of affecting agent's \mathbf{h} cognitive state (including her beliefs and intentions) in a certain way. A solution plan to an informative planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ is a sequence of informative actions $\epsilon_1, \dots, \epsilon_k$ from Op_{inf} for some k such that $\Sigma \models_{\mathbf{M}} \langle \langle \epsilon_1 \rangle \rangle \dots \langle \langle \epsilon_k \rangle \rangle \Box_{\mathbf{m}} \alpha_G$.

In an interrogative planning problem, the machine can perform both informative actions and questions. This problem is specified in the following definition.

Definition 21 (Interrogative planning problem) *An interrogative planning problem is a tuple $\langle \Sigma, Op_{\text{inf}}, Op_{\text{quest}}, \alpha_G \rangle$ where:*

- $\Sigma \subset \mathcal{L}_0$ is a finite set of agent \mathbf{m} 's available information,
- $Op_{\text{inf}} \subset Act_{\mathbf{m}}$ is a finite set of agent \mathbf{m} 's informative actions,
- $Op_{\text{quest}} \subset Que_{\mathbf{m}}$ is a finite set of agent \mathbf{m} 's questions,
- $\alpha_G \in \mathcal{L}_0$ is agent \mathbf{m} 's goal.

Intuitively, an interrogative planning problem is the problem of finding a sequence of questions as a means of understanding the interlocutor's cognitive state and, consequently, of being able to identify the inconsistencies that she must be made aware of, via a sequence of informative actions. In other words, the sequence of questions serves the purpose of "exploring" the interlocutor's cognitive state and of building a representation of it in order to being able to find a plan to reach the motivational interviewing (MI) goal.

A strong solution plan to an interrogative planning problem $\langle \Sigma, Op_{\text{inf}}, Op_{\text{quest}}, \alpha_G \rangle$ is a sequence of questions $\lambda_1, \dots, \lambda_m$ from Op_{quest} such that

$$\Sigma \models_{\mathbf{M}} \langle \langle \lambda_1 \rangle \rangle \dots \langle \langle \lambda_m \rangle \rangle \top,$$

and $\forall \rho_1 \in \mathcal{A}(\lambda_1), \dots, \forall \rho_m \in \mathcal{A}(\lambda_m), \exists \tau_1, \dots, \tau_k \in Op_{\text{inf}}$ such that

$$\Sigma \models_{\mathbf{M}} [\rho_1] \dots [\rho_m] \langle \langle \tau_1 \rangle \rangle \dots \langle \langle \tau_k \rangle \rangle \Box_{\mathbf{m}} \alpha_G.$$

A weak solution plan to an interrogative planning problem $\langle \Sigma, Op_{\text{inf}}, Op_{\text{quest}}, \alpha_G \rangle$ is a sequence of questions $\lambda_1, \dots, \lambda_m$ from Op_{quest} such that

$$\Sigma \models_{\mathbf{M}} \langle \langle \lambda_1 \rangle \rangle \dots \langle \langle \lambda_m \rangle \rangle \top,$$

and $\exists \tau_1, \dots, \tau_k \in Op_{\text{inf}}$ such that

$$\Sigma \models_{\mathbf{M}} [da(\lambda_1)] \dots [da(\lambda_m)] \langle \langle \tau_1 \rangle \rangle \dots \langle \langle \tau_k \rangle \rangle \Box_{\mathbf{m}} \alpha_G.$$

3.4 Complexity results

It is easy to verify that checking existence of a weak solution for an interrogative planning problem (EWS-INT problem) is reducible to checking existence of a solution for an informative planning problem (ES-INF problem).

As the following proposition highlights, checking existence of a solution for an informative planning problem (ES-INF problem) has the poly-size property.

Proposition 3 *An ES-INF problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan if and only if it has a poly-size solution plan $\varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$.*

SKETCH OF PROOF. It is easily seen that if an operator has been executed in a plan, another future occurrence of the same operator will not change the planning state due to the monotonicity of private belief expansion:

$$((\dots(B_i^{+\alpha})^{+\alpha_1\dots})^{+\alpha_h})^{+\alpha} = (\dots(B_i^{+\alpha})^{+\alpha_1\dots})^{+\alpha_h}.$$

■

The previous proposition is crucial for proving the following complexity result for the ES-INF problem.

Theorem 7 *The ES-INF problem is in Σ_2^P .*

SKETCH OF PROOF. By Propostion 3, an ES-INF planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan if and only if it has a poly-size solution plan. Consider a poly-time non-deterministic Turing machine with an NP-oracle (Σ_2^P -Turing machine). It begins with an empty plan and branches over all poly-size plans of length $k \leq |Op_{\text{inf}}|$ choosing non deterministically operators to add to the plan. It accepts if $\Sigma \models_{\mathbf{M}} \langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_k \rangle\rangle \Box_{\mathbf{m}} \alpha_G$ i.e., using Theorem 6, if $\neg((\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha) \rightarrow \langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_k \rangle\rangle \Box_{\mathbf{m}} \alpha_G)$ is unsatisfiable in the class \mathbf{M} . Thanks to Theorem 5, unsatisfiability of this $\mathcal{L}_{\text{Frag}}^+$ formula can be checked by the NP-oracle. When $k = |Op_{\text{inf}}|$ and the formula is satisfiable, the Turing machine rejects. ■

It is easy to verify that checking existence of a weak solution for an interrogative planning problem (EWS-INT problem) is reducible to the ES-INF problem. Thus, thanks to the previous theorem, we get the following complexity upper bound for the EWS-INT problem as a corollary.

Corollary 1 *The EWS-INT problem is in Σ_2^P .*

Checking existence of a strong solution for an interrogative planning problem (ESS-INT problem) is not comparable to the ES-INF problem or the EWS-INT problem. Indeed, it requires to take all possible answers to the questions and their possible ramifications into account. The EWS-INT problem considers a single sequence of answers (the sequence of default answers) instead.

The following theorem provides a complexity lower bound for the ES-INF problem.

Theorem 8 *The ES-INF problem is Σ_2^P -hard.*

SKETCH OF PROOF. It is well known that checking satisfiability of a $\exists\forall$ QBF is Σ_2^P -hard as it is possible to simulate an alternating Turing machine in polynomial time with 2 alternations and starting in an existential state, that decides all the problems in the class Σ_2^P . Let $\psi = \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m \varphi(x_1, \dots, x_n, y_1, \dots, y_m)$ be a quantified boolean formula (QBF) in prenex normal form. We consider the ES-INF planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ where:

$$\begin{aligned} \Sigma &= \{ \neg \Delta_{\text{h}} x_i \vee \neg \Delta_{\text{h}} \neg x_i : i \in \{1, \dots, n\} \} \\ Op_{\text{inf}} &= \{ +_{\text{m}} \Delta_{\text{h}} x_i, +_{\text{m}} \Delta_{\text{h}} \neg x_i : i \in \{1, \dots, n\} \} \\ \mathcal{P}(+_{\text{m}} \Delta_{\text{h}} x_i) &= \Diamond_{\text{m}} \Delta_{\text{h}} x_i \text{ for all } i \in \{1, \dots, n\} \\ \mathcal{P}(+_{\text{m}} \Delta_{\text{h}} \neg x_i) &= \Diamond_{\text{m}} \Delta_{\text{h}} \neg x_i \text{ for all } i \in \{1, \dots, n\} \\ \alpha_G &= \bigwedge_{i \in \{1, \dots, n\}} (\Delta_{\text{h}} x_i \vee \Delta_{\text{h}} \neg x_i) \wedge \text{encode}(\varphi(x_1, \dots, x_n, y_1, \dots, y_m)) \end{aligned}$$

where

$$\begin{aligned} \text{encode}(x_i) &= \Delta_{\text{h}} x_i \\ \text{encode}(\neg x_i) &= \Delta_{\text{h}} \neg x_i \\ \text{encode}(y_i) &= \Delta_{\text{h}} y_i \\ \text{encode}(\neg y_i) &= \neg \Delta_{\text{h}} y_i \\ \text{encode}(\varphi_1 \wedge \varphi_2) &= (\varphi_1) \wedge (\varphi_2) \\ \text{encode}(\varphi_1 \vee \varphi_2) &= (\varphi_1) \vee (\varphi_2) \end{aligned}$$

Remark that here, as Σ is consistent, it is possible to use \Diamond_{m} in the preconditions of the actions because there is at least one accessible world.

We want to prove that $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan if and only if ψ is true. (\Rightarrow) Suppose that $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan. Then by Proposition 3, it has a poly-size solution plan $P = \varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$. It is easily seen that for each $i \in \{1, \dots, n\}$, exactly one action of either $+_{\text{m}} \Delta_{\text{h}} x_i$ or $+_{\text{m}} \Delta_{\text{h}} \neg x_i$ is in the plan P . Indeed on the one hand, at most one of these actions is in the plan because $\neg \Delta_{\text{h}} x_i \vee \neg \Delta_{\text{h}} \neg x_i \in \Sigma$. And on the other hand, at least one is in the plan because of the goal $\Delta_{\text{h}} x_i \vee \Delta_{\text{h}} \neg x_i$.

(\Leftarrow) Let v be a valuation of variables in $\{x_1, \dots, x_n\}$ such that ψ is true. Hence, a solution plan for $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ is given by $\langle \langle \varepsilon_1 \rangle \dots \langle \varepsilon_n \rangle \rangle$, with for all $i \in \{1, \dots, n\}$:

$$\varepsilon_i = \begin{cases} +_{\text{m}} \Delta_{\text{h}} \neg x_i & \text{if } v(x_i) = 0 \\ +_{\text{m}} \Delta_{\text{h}} x_i & \text{if } v(x_i) = 1 \end{cases}$$

Indeed, let's prove that $\Sigma \models_{\mathbf{M}} \langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_k \rangle\rangle \Box_{\mathbf{m}} \alpha_G$ which can be also written as:

$$\Sigma \models_{\mathbf{M}} \left(\bigwedge_{i \in \{1, \dots, k\}} [+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{i-1}}] \mathcal{P}(\varepsilon_i) \right) \wedge \left([+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{k-1}}] [+_{\mathbf{m}} \alpha_{\varepsilon_k}] \Box_{\mathbf{m}} \alpha_G \right)$$

Given that, on the one hand $red([+_{\mathbf{m}} \alpha] \Diamond_{\mathbf{m}} \alpha') = red(\Diamond_{\mathbf{m}}(\alpha \wedge \alpha')) = \Diamond_{\mathbf{m}}(red(\alpha \wedge \alpha'))$, and on the other hand $red([+_{\mathbf{m}} \alpha] \Box_{\mathbf{m}} \alpha') = red(\Box_{\mathbf{m}}(\alpha \rightarrow \alpha')) = \Box_{\mathbf{m}}(red(\neg \alpha \vee \alpha'))$, when applying recursively the reduction from $\mathcal{L}_{\text{Frag}}^+$ to $\mathcal{L}_{\text{Frag}}$ we obtain:

$$\begin{aligned} red(\langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_n \rangle\rangle \Box_{\mathbf{m}} \alpha_G) &= \Diamond_{\mathbf{m}} \left(\bigwedge_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=0}} \Delta_{\mathfrak{h}} \neg x_i \right) \wedge \Diamond_{\mathbf{m}} \left(\bigwedge_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=1}} \Delta_{\mathfrak{h}} x_i \right) \\ &\quad \wedge \Box_{\mathbf{m}} \left(\left(\bigvee_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=0}} \neg \Delta_{\mathfrak{h}} \neg x_i \right) \vee \left(\bigvee_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=1}} \neg \Delta_{\mathfrak{h}} x_i \right) \vee \alpha_G \right) \end{aligned}$$

This formula can be simplified to:

$$red(\langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_n \rangle\rangle \Box_{\mathbf{m}} \alpha_G) = \Diamond_{\mathbf{m}} \left(\bigwedge_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=0}} \Delta_{\mathfrak{h}} \neg x_i \right) \wedge \Diamond_{\mathbf{m}} \left(\bigwedge_{\substack{i \in \{1, \dots, n\} \\ v(x_i)=1}} \Delta_{\mathfrak{h}} x_i \right) \wedge \Box_{\mathbf{m}}(\alpha_G)$$

■

The complexity lower bound for the EWS-INT problem follows as a corollary.

Corollary 2 *The EWS-INT problem is Σ_2^P -hard.*

3.5 Belief Revision Module

In this section, we describe the belief revision module of the architecture we sketched in Section 3.2. As we emphasized above, such a module is necessary for updating the machine's belief base after the human has replied to its questions.

Let $\mathcal{L}_{\text{PROP}}$ be the propositional language built from the following set of atomic formulas:

$$Atm^+ = Atm \cup \{p_{\Delta_i \alpha} : \Delta_i \alpha \in \mathcal{L}_0\}.$$

Moreover, let tr_{PROP} be the following translation from the language \mathcal{L}_0 defined in

Section 2.1 to $\mathcal{L}_{\text{PROP}}$:

$$\begin{aligned} tr_{\text{PROP}}(p) &= p, \\ tr_{\text{PROP}}(\neg\alpha) &= \neg tr_{\text{PROP}}(\alpha), \\ tr_{\text{PROP}}(\alpha_1 \wedge \alpha_2) &= tr_{\text{PROP}}(\alpha_1) \wedge tr_{\text{PROP}}(\alpha_2), \\ tr_{\text{PROP}}(\Delta_i\alpha) &= p_{\Delta_i\alpha}. \end{aligned}$$

For each finite $X \subseteq \mathcal{L}_0$, we define $tr_{\text{PROP}}(X) = \{tr_{\text{PROP}}(\alpha) : \alpha \in X\}$. Moreover, we say that X is propositionally consistent if and only if $\perp \notin Cn(tr_{\text{PROP}}(X))$, where Cn is the classical deductive closure operator over the propositional language $\mathcal{L}_{\text{PROP}}$. Clearly, the latter is equivalent to saying that $\bigwedge_{\alpha \in X} tr_{\text{PROP}}(\alpha)$ is satisfiable in propositional logic.

Let $\Sigma_{\text{core}}, \Sigma_{\text{mut}} \subseteq \mathcal{L}_0$ denote, respectively, the core (or, immutable) information in agent \mathbf{m} 's belief base and the volatile (or, mutable) information in agent \mathbf{m} 's belief base. Agent \mathbf{m} 's core beliefs are stable and do not change under belief revision. On the contrary, volatile beliefs can change due to a belief revision operation. Moreover, let $\Sigma_{\text{input}} \subseteq \mathcal{L}_0$ be agent \mathbf{m} 's input information set. We define $\Sigma_{\text{base}} = \Sigma_{\text{core}} \cup \Sigma_{\text{mut}}$. The revision of $(\Sigma_{\text{core}}, \Sigma_{\text{mut}})$ by input Σ_{input} , noted $Rev(\Sigma_{\text{core}}, \Sigma_{\text{mut}}, \Sigma_{\text{input}})$, is formally defined as follows:

1. if $\Sigma_{\text{core}} \cup \Sigma_{\text{input}}$ is not propositionally consistent then $Rev(\Sigma_{\text{core}}, \Sigma_{\text{mut}}, \Sigma_{\text{input}}) = (\Sigma_{\text{core}}, \Sigma_{\text{mut}})$,
2. otherwise, $Rev(\Sigma_{\text{core}}, \Sigma_{\text{mut}}, \Sigma_{\text{input}}) = (\Sigma'_{\text{core}}, \Sigma'_{\text{mut}})$, with $\Sigma'_{\text{core}} = \Sigma_{\text{core}}$ and

$$\Sigma'_{\text{mut}} = \bigcap_{X \in MCS(\Sigma_{\text{core}}, \Sigma_{\text{mut}}, \Sigma_{\text{input}})} X,$$

where $X \in MCS(\Sigma_{\text{core}}, \Sigma_{\text{mut}}, \Sigma_{\text{input}})$ if and only if:

- $X \subseteq \Sigma_{\text{mut}} \cup \Sigma_{\text{input}}$,
- $\Sigma_{\text{input}} \subseteq X$,
- $X \cup \Sigma_{\text{core}}$ is propositionally consistent, and
- there is no $X' \subseteq \Sigma_{\text{mut}} \cup \Sigma_{\text{input}}$ such that $X \subset X'$ and $X' \cup \Sigma_{\text{core}}$ is propositionally consistent.

The revision function Rev has the following effects on agent \mathbf{m} 's beliefs: (i) the core belief base is not modified, while (ii) the input Σ_{input} is added to the mutable belief base only if it is consistent with the core beliefs. If the latter is the case, then the updated mutable belief base is equal to the intersection of the subsets of the mutable belief base which are maximally consistent with respect to the core

belief base and which include the input Σ_{input} .³ This guarantees that belief revision satisfies minimal change. The function Rev is a screened revision operator as defined in [Makinson 1997]. The latter was recently generalized to the multi-agent case [Lorini & Schwarzentruher 2021]. Let $Rev(\Sigma_{core}, \Sigma_{mut}, \Sigma_{input}) = (\Sigma'_{core}, \Sigma'_{mut})$.

For notational convenience, we write $Rev^{core}(\Sigma_{core}, \Sigma_{mut}, \Sigma_{input})$ to denote Σ'_{core} and $Rev^{mut}(\Sigma_{core}, \Sigma_{mut}, \Sigma_{input})$ to denote Σ'_{mut} . Note that, if Σ_{base} is propositionally consistent, then $Rev^{core}(\Sigma_{core}, \Sigma_{mut}, \Sigma_{input}) \cup Rev^{mut}(\Sigma_{core}, \Sigma_{mut}, \Sigma_{input})$ is propositionally consistent too.

3.6 Example 1: Artificial assistant

In this section, we illustrate an example of an ES-INF problem. This example explores only one direction of the interaction between the two agents and does not use the belief revision component of the architecture we sketched in section 3.2.

We consider a HMI scenario in which agent m is the artificial assistant of the human agent h . Agent h has to choose a sport to practice since her doctor recommended her to do a regular physical activity to be in good health. Agent m 's aim is to help agent h to make the right choice, given her actual beliefs and desires. The finite set of sport activities from which h can choose is noted Opt . Elements of Opt are noted o, o', \dots . Each option in Opt is identified with a finite set of variables Var . Each variable x in Var takes a value from its corresponding finite set of values Val_x .

In this example, we suppose that Opt is composed of the following eight elements: swimming (sw), running (ru), horse riding (hr), tennis (te), soccer (so), yoga (yo), diving (di) and squash (sq). Moreover, there are exactly six variables in Var which are used to classify the available options: environment (**env**), location (**loc**), sociality (**soc**), cost (**cost**), dangerousness (**dan**) and intensity (**intens**). The set of values for the variables are:

$$\begin{aligned} Val_{\mathbf{env}} &= \{land, water\}, \\ Val_{\mathbf{loc}} &= \{indoor, outdoor, mixed\}, \\ Val_{\mathbf{soc}} &= \{single, team, mixed\}, \\ Val_{\mathbf{cost}} &= \{low, med, high\}, \\ Val_{\mathbf{dan}} &= \{low, med, high\}, \\ Val_{\mathbf{intens}} &= \{low, med, high\}. \end{aligned}$$

³Note that the revision function Rev does not expand agent m 's core belief set Σ_{core} with the input information set Σ_{input} . It would be interesting to introduce a function $f_{appr} : \mathcal{L}_0 \rightarrow \{0, 1\}$ which specifies for every formula α in \mathcal{L}_0 whether the information α is completely apprehensible by agent m (i.e., $f_{appr}(\alpha) = 1$) or not (i.e., $f_{appr}(\alpha) = 0$). Specifically, $f_{appr}(\alpha) = 1$ means that if agent m learns that α is true then, as a consequence, it will firmly believe that α is true thereby adding α not only to its set of mutable beliefs but also to its set of core beliefs. The function f_{appr} would allow us to define a variant of belief revision according to which if $\Sigma_{core} \cup \Sigma_{input}$ is propositionally consistent, then the core belief set Σ_{core} is expanded by all formulas α in Σ_{input} such that $f_{appr}(\alpha) = 1$, that is, $\Sigma'_{core} = \Sigma_{core} \cup \{\alpha \in \Sigma_{input} : f_{appr}(\alpha) = 1\}$.

The set of assignments for variable x is defined as follows:

$$Assign_x = \{x \mapsto v : v \in Val_x\}.$$

The set of variable assignments is

$$Assign = \bigcup_{x \in Var} Assign_x.$$

Elements of $Assign$ are noted a, a', \dots

We assume that the content of an atomic desire is a variable assignment or its negation. That is, agent \mathfrak{h} 's atomic desire can be any element from the following set:

$$Des_0 = Assign \cup \{\sim a : a \in Assign\}.$$

Elements of Des_0 are noted d, d', \dots . For example, the fact that \mathfrak{h} has $\mathbf{loc} \mapsto \mathit{indoor}$ as a desire means that \mathfrak{h} would like to practice an indoor activity, while if \mathfrak{h} 's desire is $\sim \mathbf{cost} \mapsto \mathit{high}$, then \mathfrak{h} would like to practice an activity whose cost is not high. Agent \mathfrak{h} 's desires are either atomic desires or conditional desires. That is, \mathfrak{h} 's desire can be any element from the following set:

$$Des = Des_0 \cup \{[d_1, \dots, d_k] \rightsquigarrow d : d_1, \dots, d_k, d \in Des_0\}.$$

Elements of Des are noted γ, γ', \dots . For example, if agent \mathfrak{h} has $[\mathbf{cost} \mapsto \mathit{high}] \rightsquigarrow \mathbf{dan} \mapsto \mathit{low}$ as a desire, then she would like to practice a sport whose dangerousness level is low, if its cost is high. We define $2^{Des^*} = 2^{Des} \setminus \emptyset$.

Let us assume that the set Atm includes four types of atomic formulas, for every $x \mapsto v \in Assign$, $o, o' \in Opt$ and $\Gamma \in 2^{Des^*}$: (i) $\mathbf{val}(o, x \mapsto v)$ standing for “option o has value v for variable x ”, (ii) $\mathbf{ideal}(\mathfrak{h}, o)$ standing for “ o is an ideal option for agent \mathfrak{h} ”, (iii) $\mathbf{justif}(\mathfrak{h}, o)$ standing for “agent \mathfrak{h} has a justification for choosing option o ”, and (iv) $\mathbf{des}(\mathfrak{h}, \Gamma)$ standing for “ Γ is agent \mathfrak{h} 's set of desires”.

The following function f_{comp} specifies, for every option $o \in Opt$ and possible desire $\gamma \in Des$, the condition guaranteeing that o satisfies (or, complies with) γ :

$$\begin{aligned} f_{comp}(o, a) &= \mathbf{val}(o, a), \\ f_{comp}(o, \sim a) &= \neg \mathbf{val}(o, a), \\ f_{comp}(o, [d_1, \dots, d_k] \rightsquigarrow d) &= \neg f_{comp}(o, d_1) \vee \dots \vee \neg f_{comp}(o, d_k) \vee f_{comp}(o, d). \end{aligned}$$

The following function $f_{comp}^{\mathfrak{h}}$ specifies, for every option $o \in Opt$ and possible desire $\gamma \in Des$, the condition guaranteeing that agent \mathfrak{h} believes that o satisfies γ :

$$\begin{aligned} f_{comp}^{\mathfrak{h}}(o, a) &= \Delta_{\mathfrak{h}} f_{comp}(o, a), \\ f_{comp}^{\mathfrak{h}}(o, \sim a) &= \Delta_{\mathfrak{h}} f_{comp}(o, \sim a), \\ f_{comp}^{\mathfrak{h}}(o, [d_1, \dots, d_k] \rightsquigarrow d) &= \Delta_{\mathfrak{h}} \neg f_{comp}(o, d_1) \vee \dots \vee \Delta_{\mathfrak{h}} \neg f_{comp}(o, d_k) \vee \Delta_{\mathfrak{h}} f_{comp}(o, d). \end{aligned}$$

The previous formulation of $f_{comp}^h(o, [d_1, \dots, d_k] \rightsquigarrow d)$ presupposes an understanding of conditional (goal) sentences by agent h . In particular, agent m does not need to provide information to agent h about the antecedent of the conditional, if the consequent is true.

We assume that the artificial agent m has the following pieces of information in its belief base:

$$\begin{aligned}
\alpha_1 &\stackrel{\text{def}}{=} \bigwedge_{\substack{o \in Opt \\ x \in Var \\ v, v' \in Val_x: v \neq v'}} (\text{val}(o, x \mapsto v) \rightarrow \neg \text{val}(o, x \mapsto v')), \\
\alpha_2 &\stackrel{\text{def}}{=} \bigwedge_{\substack{o \in Opt \\ x \in Var \\ v, v' \in Val_x: v \neq v'}} (\Delta_h \text{val}(o, x \mapsto v) \rightarrow \Delta_h \neg \text{val}(o, x \mapsto v')), \\
\alpha_3 &\stackrel{\text{def}}{=} \bigwedge_{\Gamma, \Gamma' \in 2^{Des^*}: \Gamma \neq \Gamma'} (\text{des}(h, \Gamma) \rightarrow \neg \text{des}(h, \Gamma')), \\
\alpha_4 &\stackrel{\text{def}}{=} \bigvee_{\Gamma \in 2^{Des^*}} \text{des}(h, \Gamma), \\
\alpha_5 &\stackrel{\text{def}}{=} \bigwedge_{o \in Opt} (\text{ideal}(h, o) \leftrightarrow \bigvee_{\Gamma \in 2^{Des^*}} (\text{des}(h, \Gamma) \wedge \bigwedge_{\gamma \in \Gamma} f_{comp}(o, \gamma))), \\
\alpha_6 &\stackrel{\text{def}}{=} \bigwedge_{o \in Opt} (\text{justif}(h, o) \leftrightarrow \bigvee_{\Gamma \in 2^{Des^*}} (\text{des}(h, \Gamma) \wedge \bigwedge_{\gamma \in \Gamma} f_{comp}^h(o, \gamma))).
\end{aligned}$$

Formula α_1 captures the fact that a sport cannot have two different values for a given variable. Formula α_2 is its subjective version for agent h . Formulas α_3 and α_4 capture together the fact that agent h has exactly one non-empty set of desires. According to formula α_5 , an option o is ideal for agent h if and only if it satisfies all agent h 's desires. Finally, according to formula α_6 , agent h has a reasonable justification for choosing option o if and only if she has all necessary information to conclude that option o satisfies all her desires.

We also assume that agent m has in its belief base a complete representation of Table 3.1, which specifies the variable assignments for all options:

$$\alpha_7^{o,x} \stackrel{\text{def}}{=} \text{val}(o, x \mapsto v_{o,x}).$$

In order to help agent h to select an activity, agent m also needs information about h 's set of actual desires. The latter is captured by the following formula:

$$\alpha_8 \stackrel{\text{def}}{=} \text{des}(h, \Gamma_h), \text{ with}$$

$$\begin{aligned}
\Gamma_h = \{ &\mathbf{env} \mapsto \textit{land}, \mathbf{intens} \mapsto \textit{med}, \sim \mathbf{loc} \mapsto \textit{indoor}, \\
&[\mathbf{cost} \mapsto \textit{high}] \rightsquigarrow \mathbf{soc} \mapsto \textit{mixed} \}.
\end{aligned}$$

	env	loc	soc	cost	dan	intens
sw	water	mixed	single	med	low	high
ru	land	outdoor	single	low	med	high
hr	land	outdoor	single	high	high	low
te	land	mixed	mixed	high	med	med
so	land	mixed	team	med	med	med
yo	land	mixed	single	med	low	low
di	water	mixed	single	high	high	low
sq	land	indoor	mixed	high	med	med

Table 3.1: Variable assignments. For every option $o \in Opt$ and variable $x \in Var$, we denote by $v_{o,x}$ the corresponding entry in the table. For instance, we have $v_{sw,env} = water$.

This means that, according to agent m , agent h would like to practice a land activity, with medium intensity, which is not exclusively indoor, and which can be practiced both in single and team mode, if its cost is high.

Let us now turn to the informative planning problem. We suppose agent m 's set of operators Op_{inf} is:

$$Op_{inf} = \{ convince(m, h, val(o, a)) : o \in Opt \text{ and } a \in Assign \} \cup \{ convince(m, h, ideal(h, o)) : o \in Opt \}.$$

In other words, agent m can only inform agent h about an option's value for a certain variable or about the ideality of an option for her.

We use the speech act *convince* since we suppose agent h fully trusts what agent m says (i.e., h believes that m is both sincere and competent).

We suppose the following executability precondition for every $o \in Opt$ and $a \in Assign$:

$$\mathcal{P}(convince(m, h, val(o, a))) = \Box_m \left(val(o, a) \wedge \bigwedge_{v \in Val_{dan}} (val(o, dan \mapsto v) \rightarrow \Delta_h val(o, dan \mapsto v)) \right) \\ \text{if } a \notin Assign_{dan},$$

$$\mathcal{P}(convince(m, h, val(o, a))) = \Box_m val(o, a) \\ \text{if } a \in Assign_{dan},$$

$$\mathcal{P}(convince(m, h, ideal(h, o))) = \Box_m (ideal(h, o) \wedge justif(h, o)).$$

According to the first definition, agent m can inform agent h about an option's value for a certain variable, if and only if this information is believed by m and m believes that h has been already informed about the dangerousness level of the option. Indeed, we assume that, before being presented with an option's features, agent h must be informed about its the dangerousness level and agent m complies with this rule. The second definition simply stipulates that m can inform h about the dangerousness level of an option if and only if it believes what it says. Finally, according to the third definition, m can inform h about the ideality of an option only if it believes that h has a reasonable justification for choosing it. Indeed, we assume

\mathbf{m} will inform \mathbf{h} about the ideality of an option only after having explained why the option is ideal for her. The three definitions presuppose that agent \mathbf{m} cannot spread fake news (i.e., something that it does not implicitly believe).

We moreover suppose that, for agent \mathbf{h} to have a potential intention to choose option o , denoted by $\text{potIntend}(\mathbf{h}, o)$, she must have a justified belief that o is an ideal option for her:⁴

$$\text{potIntend}(\mathbf{h}, o) \stackrel{\text{def}}{=} \Delta_{\mathbf{h}}\text{ideal}(\mathbf{h}, o) \wedge \text{justif}(\mathbf{h}, o).$$

This abbreviation together with the abbreviation α_6 given above relate intention with belief and desire, in line with existing theories of intention [Audi 1973, Davidson 1980].

It turns out that the sequence of speech acts $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ with

$$\begin{aligned} \epsilon_1 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(\text{te}, \mathbf{dan} \mapsto \text{med})), \\ \epsilon_2 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(\text{te}, \mathbf{env} \mapsto \text{land})), \\ \epsilon_3 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(\text{te}, \mathbf{intens} \mapsto \text{med})), \\ \epsilon_4 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(\text{te}, \mathbf{loc} \mapsto \text{mixed})), \\ \epsilon_5 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(\text{te}, \mathbf{soc} \mapsto \text{mixed})), \\ \epsilon_6 &\stackrel{\text{def}}{=} \text{convince}(\mathbf{m}, \mathbf{h}, \text{ideal}(\mathbf{h}, \text{te})). \end{aligned}$$

provides a solution for the informative planning problem $\langle \Sigma, \text{Op}_{\text{inf}}, \alpha_G \rangle$, where

$$\Sigma = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_8\} \cup \{\alpha_7^{o,x} : o \in \text{Opt} \text{ and } x \in \text{Var}\},$$

Op_{inf} has the previous specifications and agent \mathbf{h} 's persuasive goal α_G is defined as follows:

$$\alpha_G \stackrel{\text{def}}{=} \bigvee_{o \in \text{Opt}} \text{potIntend}(\mathbf{h}, o).$$

This means that, by performing the sequence of operators $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$, agent \mathbf{m} will induce agent \mathbf{h} to form a potential intention to choose an activity. In other words, agent \mathbf{m} will provide an effective recommendation to agent \mathbf{h} .

We conclude this section with a general observation about the formulation of the planning problem for our example. Let $\langle \Sigma, \text{Op}_{\text{inf}}, \alpha_G \rangle$ be the informative planning problem we want to solve. Let \mathbf{m} 's set of operators for option $o \in \text{Opt}$ relative to

⁴Our account of potential intention is reminiscent of the JTB ('justified true belief') account to knowledge [Goldman 1979].

$\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ be defined as follows:

$$Op_o^{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle} = \{ \text{convince}(\mathbf{m}, \mathbf{h}, \text{val}(o, a)) : \text{val}(o, a) \in \Sigma \} \cup \{ \text{convince}(\mathbf{m}, \mathbf{h}, \text{ideal}(\mathbf{h}, o)) \}.$$

It is easy to verify that the informative planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution if and only if there exists $o \in Opt$ such that the informative planning problem $\langle \Sigma, Op_o^{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}, \alpha_G \rangle$ has a solution. Therefore, in order to solve the informative planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$, we simply need to linearly order the options in Opt and solve the informative planning problems $\langle \Sigma, Op_o^{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}, \alpha_G \rangle$ in sequence one after the other according to the ordering.

3.7 Example 2: Virtual coaching agent

In this section, we illustrate an example of an EWS-INT problem. Unlike the example of Section 3.6, this example explores both directions of the interaction between agent \mathbf{h} and agent \mathbf{m} . It moreover exploits the belief revision component of our architecture.

We illustrate the use of the cognitive planning and belief revision module of the architecture with the aid of a human-machine interaction (HMI) scenario. We assume \mathbf{m} is a virtual coaching agent which has to motivate the human agent \mathbf{h} to practice a physical activity. We suppose agent \mathbf{m} complies with the general principles of the theory of motivational interviewing (MI) to find a persuasive strategy aimed at changing the human's attitude.

3.7.1 Motivational interviewing

Motivational interviewing (for short MI) is a counseling method used in clinical psychology for eliciting behavior change [Lundahl & Burke 2009]. One crucial aspect of MI consists of exploring the participant's subjectivity through open questions to identify her desires and personal values (*e.g.*, conformity, independence, carefulness, etc.) [Miller & Rollnick 2012]. This exploration allows to become aware of the inconsistency between her desires or personal values (*e.g.*, being in good health), and her current behavior (*e.g.*, not doing enough physical activity). However, MI does not necessarily try to induce beliefs about positive aspects of the behavior change (*e.g.*, most people are already aware that reasonable physical activity is good for health and would like to practice a sport regularly). It rather helps the participant to identify the reasons why she did not convert her mere desires (*e.g.*, I would like to practice a sport) into intentions (*e.g.*, I commit to do sport regularly) and reassures her that these limitations can be overcome. To this aim, the counselor rephrases the ideas expressed by the participant so as to provoke reflections about the connection between her beliefs, desires and intentions.

Several automated MI systems have been proposed in recent times [da Silva *et al.* 2018, Kanaoka & Mutlu 2015, Lisetti *et al.* 2013,

[Olafsson *et al.* 2019, Schulman *et al.* 2011]. However, all these systems use predefined dialogue trees to conduct the MI. In this paper, we propose a model based on cognitive planning for driving MI in a human-agent interaction system.

MI is composed of several stages: prior to having the participant change her intentions, one has to make her aware of the inconsistencies between her desires and her actual behavior. In other words, she has to recognize the fact that her current behavior will prevent her from obtaining what she wants. In order to that, the artificial agent has both (i) a model of the human’s overall cognitive state, including her beliefs and intentions, and (ii) a goal towards the human’s mental attitudes, *e.g.*, the goal of making the human aware of the inconsistency between her desires and her actual behavior. Given (i) and (ii), it tries to find a sequence of speech acts aimed at modifying the human’s cognitive state thereby guaranteeing the achievement of its goal.

3.7.2 Formalization

Let us assume the disjoint sets $CondAtm$, $DesAtm$ and $ActAtm$ are subsets of the set of atomic propositions Atm . Elements of $CondAtm$ are atoms specifying conditions, while elements of $DesAtm$ are atoms specifying desirable properties, that is, properties that agent h may wish to achieve (i.e., agent h ’s possible desiderata). Finally, atoms in $ActAtm$ are used to describe agent h ’s behavior. Specifically, we define $ActAtm = \{\text{does}(h,a) : a \in Act\}$, where Act is a finite a set of action names. The atom $\text{does}(h,a)$ has to be read “agent h behaves in conformity with the requirement a ” or, simply, “agent h does action a ”.

The sets of literals from $CondAtm$, $DesAtm$ and $ActAtm$ are defined in the usual way as follows:

$$\begin{aligned} DesLit &= DesAtm \cup \{\neg p : p \in DesAtm\}, \\ CondLit &= CondAtm \cup \{\neg p : p \in CondAtm\}, \\ ActLit &= ActAtm \cup \{\neg p : p \in ActAtm\}, \\ Lit &= DesLit \cup CondLit \cup ActLit. \end{aligned}$$

We define $LitSet = 2^{Lit}$ and $LitSet_0 = LitSet \setminus \{\emptyset\}$.

We moreover assume that the set of atomic propositions Atm includes one atom $\text{des}(h,l)$ for each $l \in DesLit$ standing for “agent h desires l to be true”.

For the sake of illustration, we suppose that $Act = \{ps\}$ where ps is the action (or requirement) “to practice regularly a sport or physical activity”. Therefore, $ActAtm = \{\text{does}(h,ps)\}$. Moreover, $DesAtm = \{dr, pw, lw, at, gh, st\}$ and $CondAtm = \{ow, sl, co\}$, with the atoms having the following intuitive meaning: dr : “agent h has dietary restrictions”; pw : “agent h puts on weight”; lw : “agent h loses weight”; at : “agent h is attractive”; gh : “agent h is in good health”; st : “agent h is stressed”; ow : “agent h has an office work”; sl : “agent h has a sedentary life style”; co : “agent h is a commuter and spends quite some time in the traffic everyday”.

The following abbreviation captures a simple notion of necessity for $X \in LitSet$

and $l \in Lit$:

$$\text{nec}(X, l) \stackrel{\text{def}}{=} \bigwedge_{l' \in X} (l' \rightarrow l).$$

$\text{nec}(X, l)$ has to be read “the facts in X will not be true unless l is true” or more shortly “ l is necessary for X ”.

Agent \mathbf{m} 's initial knowledge about agent \mathbf{h} 's cognitive state is specified by the following six abbreviations:

$$\begin{aligned} \alpha_1 &\stackrel{\text{def}}{=} \bigwedge_{l \in Lit} \Delta_{\mathbf{h}} \text{nec}(\{l\}, l), \\ \alpha_2 &\stackrel{\text{def}}{=} \bigwedge_{l \in Lit} (\Delta_{\mathbf{h}} \text{nec}(\emptyset, l) \leftrightarrow \Delta_{\mathbf{h}} l), \\ \alpha_3 &\stackrel{\text{def}}{=} \bigwedge_{l \in Lit, X, X' \in LitSet: X' \subseteq X} \left(\Delta_{\mathbf{h}} \text{nec}(X, l) \rightarrow \bigwedge_{l' \in X'} (\Delta_{\mathbf{h}} l' \rightarrow \Delta_{\mathbf{h}} \text{nec}(X \setminus X', l)) \right), \\ \alpha_4 &\stackrel{\text{def}}{=} \bigwedge_{l \in Lit, X, X' \in LitSet: X \subseteq X'} (\Delta_{\mathbf{h}} \text{nec}(X, l) \rightarrow \Delta_{\mathbf{h}} \text{nec}(X', l)), \\ \alpha_5 &\stackrel{\text{def}}{=} \bigwedge_{l \in Lit} \left((\text{des}(\mathbf{h}, l) \leftrightarrow \Delta_{\mathbf{h}} \text{des}(\mathbf{h}, l)) \wedge (\neg \text{des}(\mathbf{h}, l) \leftrightarrow \Delta_{\mathbf{h}} \neg \text{des}(\mathbf{h}, l)) \right), \\ \alpha_6 &\stackrel{\text{def}}{=} \bigwedge_{a \in Act} \left((\text{does}(\mathbf{h}, a) \leftrightarrow \Delta_{\mathbf{h}} \text{does}(\mathbf{h}, a)) \wedge (\neg \text{does}(\mathbf{h}, a) \leftrightarrow \Delta_{\mathbf{h}} \neg \text{does}(\mathbf{h}, a)) \right), \end{aligned}$$

Hypotheses α_1 - α_4 are general properties about agent \mathbf{h} 's conception of necessity. According to α_1 , agent \mathbf{h} believes that every fact is necessary for itself while, according to α_2 , agent \mathbf{h} believes a fact is true regardless of the circumstances if and only if she believes that it is true. According to α_3 , if agent \mathbf{h} believes that l is necessary for the facts in X being true and believes every fact in $X' \subseteq X$, then she believes that l is necessary for the facts in the remaining set $X \setminus X'$ being true. According to α_4 , if $X \subseteq X'$ and agent \mathbf{h} believes that l is necessary for X then she believes that l is necessary for X' as well. Hypotheses α_5 and α_6 capture agent \mathbf{h} 's introspection over her desires (hypothothesis α_5) and agent \mathbf{h} 's perfect knowledge about her actions and inactions (hypothothesis α_6).

We moreover suppose that agent \mathbf{m} has the following information in its belief base capturing the necessity relations between conditions, desirable properties and actions:

$$\begin{aligned} \alpha_7 &\stackrel{\text{def}}{=} \text{nec}(\{\neg dr, \neg pw, sl\}, \text{does}(\mathbf{h}, ps)) \wedge \\ &\quad \text{nec}(\{at, \neg dr\}, \text{does}(\mathbf{h}, ps)) \wedge \\ &\quad \text{nec}(\{sl, gh\}, \text{does}(\mathbf{h}, ps)) \wedge \\ &\quad \text{nec}(\{gh\}, \neg st) \wedge \\ &\quad \text{nec}(\{co, ow\}, sl). \end{aligned}$$

For example, $\text{nec}(\{\neg dr, \neg pw, sl\}, \text{does}(\mathfrak{h}, ps))$ means that practicing regularly a sport is necessary for not having dietary restrictions and not putting weight, while having a sedentary work style (i.e., a person cannot pretend to not put weight and not have dietary restrictions without practicing a sport, if she has a sedentary work style).

The following abbreviation defines the concept of agent \mathfrak{h} 's awareness of the inconsistency between the actual state of affairs α and her desires:

$$\text{AwareIncon}(\mathfrak{h}, \alpha) \stackrel{\text{def}}{=} \bigvee_{X \in \text{LitSet}} \left(\bigwedge_{l' \in X} \text{des}(\mathfrak{h}, l') \wedge \Delta_{\mathfrak{h}} \text{nec}(X, \neg \alpha) \wedge \Delta_{\mathfrak{h}} \alpha \right).$$

According to the previous definition, agent \mathfrak{h} is aware of the inconsistency between the actual state of affairs α and her desires, noted $\text{AwareIncon}(\mathfrak{h}, \alpha)$, if she believes that the satisfaction of her desires is jeopardized by the fact that α is true. More precisely, (i) agent \mathfrak{h} believes that she will not achieve her desires unless α is false and (ii) she believes that α is actually true.

We suppose that the pieces of information $\alpha_1, \dots, \alpha_7$ constitute agent \mathfrak{m} 's initial core belief base, that is, $\Sigma_{\text{core}} = \{\alpha_1, \dots, \alpha_7\}$. Moreover, we suppose that agent \mathfrak{m} 's initial mutable belief base is empty, that is, $\Sigma_{\text{mut}} = \emptyset$. We consider the planning problem in which agent \mathfrak{m} tries to motivate agent \mathfrak{h} to practice regularly a sport. To this aim, agent \mathfrak{m} tries to achieve the following goal:

$$\alpha_G \stackrel{\text{def}}{=} \neg \text{does}(\mathfrak{h}, ps) \rightarrow \text{AwareIncon}(\mathfrak{h}, \neg \text{does}(\mathfrak{h}, ps)).$$

In other words, agent \mathfrak{m} tries to make it the case that if agent \mathfrak{h} does not practice a sport, then she becomes aware of the inconsistency between her actual desires and the fact that she does not practice a sport.

Let $X \subseteq \text{DesLit}$, $X' \subseteq \text{CondLit}$ and $l \in \text{ActLit} \cup \text{CondLit}$. We assume agent \mathfrak{m} 's action $\text{convince}(\mathfrak{m}, \mathfrak{h}, \text{nec}(X \cup X', l))$ to be concretely realized through the utterance “since condition X' holds, you will not satisfy your desires X unless l is true!”. For example, $\text{convince}(\mathfrak{m}, \mathfrak{h}, \text{nec}(X \cup X', \text{does}(\mathfrak{h}, ps)))$ corresponds to the utterance “since condition X' holds, you will not satisfy your desires X unless you do action $ps!$ ”, while $\text{convince}(\mathfrak{m}, \mathfrak{h}, \text{nec}(X \cup X', \neg \text{does}(\mathfrak{h}, ps)))$ corresponds to the utterance “since condition X' holds, you will not satisfy your desires X unless you refrain from doing action $ps!$ ”. For notational convenience, we abbreviate $\text{convince}(\mathfrak{m}, \mathfrak{h}, \text{nec}(X \cup X', l))$ by $!_{\mathfrak{m}, \mathfrak{h}}(X, X', l)$. We assume the following repertoires of informative and interrogative actions for agent \mathfrak{m} :

$$\begin{aligned} Op_{\text{inf}} &= \{!_{\mathfrak{m}, \mathfrak{h}}(X, X', l) : X \subseteq \text{DesLit}, X' \subseteq \text{CondLit} \text{ and } l \in \text{ActLit} \cup \text{CondLit}\}, \\ Op_{\text{quest}} &= \{?_{\mathfrak{m}, \mathfrak{h}} \text{des}(\mathfrak{h}, l) : l \in \text{DesLit}\} \cup \{?_{\mathfrak{m}, \mathfrak{h}} l : l \in \text{ActLit} \cup \text{CondLit}\}, \end{aligned}$$

with the following executability preconditions for their elements:

$$\begin{aligned}\mathcal{P}(!_{m,h}(X,X',l)) &= \Box_m (\text{nec}(X \cup X', l) \wedge \bigwedge_{l' \in X} \text{des}(h, l') \wedge \bigwedge_{l'' \in X'} \Delta_h l''), \\ \mathcal{P}(?_{m,h} \text{des}(h, l)) &= \mathcal{P}(?_{m,h} l) = \top.\end{aligned}$$

In other words, a question is always executable. Moreover, agent m can perform the action $!_{m,h}(X,X',l)$ — i.e., “since condition X' holds, you will not satisfy your desires X unless l is true!” — only if (i) it believes that agent h desires every fact in X to be true, (ii) it believes that agent h believes every fact in X' , and (iii) it believes that l is necessary for X when X' holds. Thus, by performing the speech act $!_{m,h}(X,X',l)$, agent m informs agent h that, in view of the fact that condition X' holds, l is necessary for the satisfaction of her desires X , since it presupposes that agent h has indeed such desires and believes that the condition holds.

We suppose that at every step k of the interaction with agent h , agent m tries to find a solution for the informative planning problem $\langle \Sigma_{base}^k, Op_{inf}^k, \alpha_G \rangle$. If it can find it, it proceeds with its execution and then interaction stops. Otherwise, it tries to find a weak solution for the interrogative planning problem $\langle \Sigma_{base}^k, Op_{inf}^k, Op_{quest}^k, \alpha_G \rangle$. If it cannot find it, the interaction stops. Otherwise, it executes the corresponding sequence of questions and revises its belief base according to agent h 's set of responses $Resp_h^k$. Then, it moves to step $k + 1$. We suppose that $\Sigma_{core}^0 = \Sigma_{core}$, $\Sigma_{mut}^0 = \Sigma_{mut}$, $Op_{inf}^0 = Op_{inf}$ and $Op_{quest}^0 = Op_{quest}$. Moreover,

$$\begin{aligned}\Sigma_{core}^{k+1} &= Rev^{core}(\Sigma_{core}^k, \Sigma_{mut}^k, Resp_h^k), \\ \Sigma_{mut}^{k+1} &= Rev^{mut}(\Sigma_{core}^k, \Sigma_{mut}^k, Resp_h^k), \\ Op_{inf}^{k+1} &= Op_{inf}^k, \\ Op_{quest}^{k+1} &= Op_{quest}^k \setminus Selected(Op_{quest}^k),\end{aligned}$$

where $Selected(Op_{quest}^k)$ is the set of questions included in the interrogative plan selected at step k . We remove them because we want to avoid that agent m keeps asking the same question indefinitely.

Let us illustrate an example of interaction. At step 0, agent m cannot find a solution for the informative planning problem. Thus, it decides to go with questions. It finds $?_{m,h} \text{does}(h, ps)$ as solution for the interrogative planning problem. We suppose agent h 's response to agent m 's question is $+_m \neg \Delta_h \text{does}(h, ps)$. At step 1, again agent m cannot find a solution for the informative planning problem. Thus, it moves to the interrogative planning problem and finds the following sequence of questions as a weak solution:

$$?_{m,h} \text{des}(h, gh), ?_{m,h} co, ?_{m,h} ow.$$

Agent m executes the interrogative plan. We suppose agent h 's set of responses to agent m 's questions at step 1 is $\{+_m \Delta_h \text{des}(h, gh), +_m \Delta_h co, +_m \Delta_h ow\}$.

Thus, at step 2, agent m can find a solution for the informative planning problem. The solution is the following sequence of assertive speech acts of length 2:

$$!_{m,h}(\emptyset, \{co, ow\}, sl), !_{m,h}(\{gh\}, \{sl\}, \text{does}(h, ps)).$$

Agent m executes the informative plan. The previous interaction between agent m and agent h is illustrated in Figure 3.2 in which every speech act is associated with its corresponding utterance.

Speaker	Utterance	Speech act
m	Do you practice a sport regularly?	? _{m,h} does(h,ps)
h	I don't	+ _m ¬ _h des(h,ps)
m	Do you wish to be in good health?	? _{m,h} des(h,gh)
h	Yes	+ _m △ _h des(h,gh)
m	Do you spend quite some time in the traffic everyday as a commuter?	? _{m,h} co
h	Yes	+ _m △ _h co
m	Do you have an office work?	? _{m,h} ow
h	Yes	+ _m △ _h ow
m	You spend quite some time in the traffic everyday as a commuter and you have an office work. Therefore, your life style is sedentary!	! _{m,h} (∅, {co, ow}, sl)
m	Your life style is sedentary. Therefore, you will not satisfy your desire to be in good health unless you practice a sport regularly!	! _{m,h} ({gh}, {sl}, does(h,ps))

Figure 3.2: Human-machine dialogue

3.8 Optimal QBF Encoding

We present now a QBF encoding of the ES-INF problem with prefix $\exists\forall$. Intuitively, a solution plan candidate is non-deterministically chosen (\exists) and the validity of this plan is checked (\forall). Formally, we introduce global variables (atomic propositions) whose truth value are independent of the beliefs of the agents and therefore is the same in all mental states. These variables will be used as *selectors* in a $\mathcal{L}_{\text{Frag}}$ formula in order to focus on the reduction of different $\mathcal{L}_{\text{Frag}}^+$ formulas depending on the truth values of these global variables.

For a given ES-INF planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$, by Proposition 3, we can only consider poly-size solution plan candidates of the form $\varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$. We define the set of global selector variables $V_s = \{s_{\varepsilon \preceq \varepsilon'} : \varepsilon, \varepsilon' \in Op_{\text{inf}}\} \subseteq \text{Atm}$ and the formula φ_{V_s} as the conjunction of the following axioms:

$$\bigwedge_{\varepsilon \in Op_{\text{inf}}} \left(\neg s_{\varepsilon} \preceq \varepsilon \right) \quad \rightarrow \quad \bigwedge_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} \left(\neg s_{\varepsilon} \preceq \varepsilon' \wedge \neg s_{\varepsilon'} \preceq \varepsilon \right) \quad (\text{S1})$$

$$\bigwedge_{\varepsilon \in Op_{\text{inf}}} \bigwedge_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} \left(\neg s_{\varepsilon} \preceq \varepsilon' \vee \neg s_{\varepsilon'} \preceq \varepsilon \right) \quad (\text{S2})$$

$$\bigwedge_{\varepsilon \in Op_{\text{inf}}} \bigwedge_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} \bigwedge_{\substack{\varepsilon'' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'' \\ \varepsilon' \neq \varepsilon''}} \left(s_{\varepsilon} \preceq \varepsilon' \wedge s_{\varepsilon'} \preceq \varepsilon'' \rightarrow s_{\varepsilon} \preceq \varepsilon'' \right) \quad (\text{S3})$$

$$\bigwedge_{\varepsilon \in Op_{\text{inf}}} \bigwedge_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} \left(s_{\varepsilon} \preceq \varepsilon \wedge s_{\varepsilon'} \preceq \varepsilon' \rightarrow s_{\varepsilon} \preceq \varepsilon' \vee s_{\varepsilon'} \preceq \varepsilon \right) \quad (\text{S4})$$

$$\bigwedge_{\varepsilon \in Op_{\text{inf}}} \bigwedge_{\varepsilon' \in Op_{\text{inf}}} \left(s_{\varepsilon} \preceq \varepsilon' \leftrightarrow \Box_{\mathbf{m}} s_{\varepsilon} \preceq \varepsilon' \right) \quad (\text{S5})$$

These axioms are constructed in order to set a bijective function between the models of φ_{V_s} and the solution plan candidates for $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$. In the sequel, for a given model of φ_{V_s} , the corresponding plan will be called the designated plan. Intuitively, we define a total order \preceq between the elements of the designated plan, and other actions from Op_{inf} are not ordered at all. Axiom S1 states that if an action $\varepsilon \in Op_{\text{inf}}$ is not selected in the designated plan then no other action $\varepsilon' \in Op_{\text{inf}}$ is \preceq -related with ε . Note that its contrapositive implies the reflexivity of \preceq on elements of the designated plan and only on these elements of Op_{inf} . The following axioms define antisymmetry (S2) and transitivity (S3) of \preceq on elements of the designated plan. Finally, axiom S4 states that \preceq is total on elements of the designated plan.

A plan candidate $\varepsilon_1, \dots, \varepsilon_k$ is a solution plan for $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ if and only if the following formula is valid:

$$\left(\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha \right) \rightarrow \langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_{k-1} \rangle\rangle \langle\langle \varepsilon_k \rangle\rangle \Box_{\mathbf{m}} \alpha_G$$

which can be also written as:

$$\left(\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha \right) \rightarrow \left(\bigwedge_{i \in \{1, \dots, k\}} [+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{i-1}}] \mathcal{P}(\varepsilon_i) \right) \\ \wedge [+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{k-1}}] [+_{\mathbf{m}} \alpha_{\varepsilon_k}] \Box_{\mathbf{m}} \alpha_G$$

Remark that for a given $i \in \{1, \dots, k\}$, we have

$$\text{red}([+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{i-1}}] \mathcal{P}(\varepsilon_i)) = \begin{cases} \top, & \text{if } \mathcal{P}(\varepsilon_i) = \Delta_{\mathbf{m}} \alpha_{\varepsilon_j} \text{ for some } j < i, \\ \text{red}(\mathcal{P}(\varepsilon_i)), & \text{otherwise;} \end{cases}$$

In words, if the precondition $\mathcal{P}(\varepsilon_i)$ of an operator ε_i is an explicit belief $\Delta_{\mathbf{m}} \alpha_{\varepsilon_j}$ of agent \mathbf{m} which is added by another operator ε_j which precedes ε_i in the plan candidate, then the reduction of the test of the precondition of ε_i into $\mathcal{L}_{\text{Frag}}$ is set to \top . Hence, we can use a selector variable $s_{\varepsilon_j \preceq \varepsilon_i}$ to generate a reduction depending on its value by replacing \top by $s_{\varepsilon_j \preceq \varepsilon_i} \vee \Delta_{\mathbf{m}} \alpha_{\varepsilon_j}$. Indeed, if ε_j is selected and precedes ε_i in a designated plan then $s_{\varepsilon_j \preceq \varepsilon_i}$ and the reduction are equivalent to \top , else $s_{\varepsilon_j \preceq \varepsilon_i}$ is equivalent to \perp and the reduction is equivalent to $\Delta_{\mathbf{m}} \alpha_{\varepsilon_j}$.

Moreover, if $\mathcal{P}(\varepsilon_i)$ contains implicit belief subformulas of the form $\Box_{\mathbf{m}} \alpha$, the reduction of such a subformula is given by:

$$\text{red}([+_{\mathbf{m}} \alpha_{\varepsilon_1}] \dots [+_{\mathbf{m}} \alpha_{\varepsilon_{i-1}}] \Box_{\mathbf{m}} \alpha) = \Box_{\mathbf{m}} (\alpha_{\varepsilon_1} \rightarrow \dots \rightarrow \alpha_{\varepsilon_{i-1}} \rightarrow \alpha) \\ = \Box_{\mathbf{m}} \left(\left(\bigvee_{j \in \{1, \dots, i-1\}} \neg \alpha_{\varepsilon_j} \right) \vee \alpha \right)$$

As previously, we can use a selector variable $s_{\varepsilon_j \preceq \varepsilon_i}$ to generate a reduction depending on its value by replacing $\neg \alpha_{\varepsilon_j}$ by $s_{\varepsilon_j \preceq \varepsilon_i} \wedge \neg \alpha_{\varepsilon_j}$. Indeed, if ε_j is selected and precedes ε_i in a designated plan then $s_{\varepsilon_j \preceq \varepsilon_i}$ is equivalent to \top and the disjunct $\neg \alpha_{\varepsilon_j}$ is present in the reduction, else $s_{\varepsilon_j \preceq \varepsilon_i}$ is equivalent to \perp and the disjunct $\neg \alpha_{\varepsilon_j}$ is absent from the reduction.

Finally, we can then define a function generating the reduction of the precondition $\mathcal{P}(\varepsilon)$ of any action ε from a designated plan, depending on all the actions ε' that precede ε in the designated plan given by a model of φ_{V_s} . For $\varepsilon \in \text{Op}_{\text{inf}}$, we define such a function $\Pi_{\preceq \varepsilon} : \mathcal{L}_{\text{Frag}} \rightarrow \mathcal{L}_{\text{Frag}}$ such that:

$$\begin{aligned}
\Pi_{\prec\varepsilon}(p) &= p, \\
\Pi_{\prec\varepsilon}(\alpha) &= \begin{cases} s_{\varepsilon' \prec \varepsilon} \vee \Delta_{\mathbf{m}} \alpha_{\varepsilon'}, & \text{if } \exists \varepsilon' \neq \varepsilon : \alpha = \Delta_{\mathbf{m}} \alpha_{\varepsilon'}, \\ \alpha, & \text{otherwise;} \end{cases} \\
\Pi_{\prec\varepsilon}(\neg\varphi) &= \neg\Pi_{\prec\varepsilon}(\varphi), \\
\Pi_{\prec\varepsilon}(\varphi_1 \wedge \varphi_2) &= \Pi_{\prec\varepsilon}(\varphi_1) \wedge \Pi_{\prec\varepsilon}(\varphi_2), \\
\Pi_{\prec\varepsilon}(\varphi_1 \vee \varphi_2) &= \Pi_{\prec\varepsilon}(\varphi_1) \vee \Pi_{\prec\varepsilon}(\varphi_2), \\
\Pi_{\prec\varepsilon}(\Box_{\mathbf{m}} \alpha) &= \Box_{\mathbf{m}} \left(\left(\bigvee_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} s_{\varepsilon' \prec \varepsilon} \wedge \neg \alpha_{\varepsilon'} \right) \vee \alpha \right) \\
\Pi_{\prec\varepsilon}(\Diamond_{\mathbf{m}} \alpha) &= \Diamond_{\mathbf{m}} \left(\bigwedge_{\substack{\varepsilon' \in Op_{\text{inf}} \\ \varepsilon \neq \varepsilon'}} (\neg s_{\varepsilon' \prec \varepsilon} \vee \alpha_{\varepsilon'}) \wedge \alpha \right)
\end{aligned}$$

For the goal, we proceed in a similar manner to calculate the reduction depending on the selector variables, and we obtain the formula:

$$\varphi_P = \Box_{\mathbf{m}} \left(\left(\bigvee_{\varepsilon \in Op_{\text{inf}}} s_{\varepsilon \prec \varepsilon} \wedge \neg \alpha_{\varepsilon} \right) \vee \alpha_G \right)$$

It is now easily seen that, depending on the values of selector variables, the following formula of $\mathcal{L}_{\text{Frag}}$ represents all possible formulas that allows us to check the validity of all plan candidates for the planning problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$:

$$\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle} = \varphi_{V_s} \wedge \left(\left(\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha \right) \rightarrow \bigwedge_{\varepsilon \in Op_{\text{inf}}} \left(s_{\varepsilon \prec \varepsilon} \rightarrow \Pi_{\prec\varepsilon}(\mathcal{P}(\varepsilon)) \right) \wedge \varphi_P \right)$$

Indeed,

Proposition 4 *Given a plan candidate $P = \varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$, if we consider the valuation v of variables in V_s such that P is the corresponding designated plan, then the formula $\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ is equivalent to*

$$\text{red} \left(\left(\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha \right) \rightarrow \langle\langle \varepsilon_1 \rangle\rangle \dots \langle\langle \varepsilon_{k-1} \rangle\rangle \langle\langle \varepsilon_k \rangle\rangle \Box_{\mathbf{m}} \alpha_G \right)$$

SKETCH OF PROOF. Let $P = \varepsilon_1, \dots, \varepsilon_k$ a sequence of actions such that $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$. We consider the valuation v of variables in V_s such that

P is the corresponding designated plan. Then, v is a model of φ_{V_s} , and $\forall i, j$ such that $1 \leq i \leq j \leq k$, we have $v \models s_{\varepsilon_i \preceq \varepsilon_j}$ and $\forall \varepsilon, \varepsilon' \in Op_{\text{inf}} \setminus P$ we have $v \not\models s_{\varepsilon \preceq \varepsilon'}$. Hence we can simplify the formula:

$$\left(\left(\bigwedge_{\alpha \in \Sigma} \square_m \alpha \right) \rightarrow \bigwedge_{\varepsilon \in Op_{\text{inf}}} \left(s_{\varepsilon \preceq \varepsilon} \rightarrow \Pi_{\preceq \varepsilon}(\mathcal{P}(\varepsilon)) \right) \right) \wedge \square_m \left(\left(\bigvee_{\varepsilon \in Op_{\text{inf}}} s_{\varepsilon \preceq \varepsilon} \wedge \neg \alpha_\varepsilon \right) \vee \alpha_G \right)$$

$$\left(\left(\bigwedge_{\alpha \in \Sigma} \square_m \alpha \right) \rightarrow \bigwedge_{\varepsilon \in Op_{\text{inf}}} \Pi_{\preceq \varepsilon}(\mathcal{P}(\varepsilon)) \right) \wedge \square_m \left(\left(\bigvee_{\varepsilon \in Op_{\text{inf}}} \neg \alpha_\varepsilon \right) \vee \alpha_G \right)$$

Moreover, by construction of the function $\Pi_{\preceq \varepsilon}$, we know for each $i \in \{1, \dots, k\}$ with $\varepsilon = \varepsilon_i$ that $\Pi_{\preceq \varepsilon}(\mathcal{P}(\varepsilon))$ is the reduction of $[+_m \alpha_{\varepsilon_1}] \dots [+_m \alpha_{\varepsilon_{i-1}}] \mathcal{P}(\varepsilon)$ for the designated plan P . At this point, there is no more selector variables in the simplified formula and we obtain the result by definition of the reduction function. ■

In order to propagate the selector variables the right way, we extend the definitions of translations tr_0 , tr_1 and tr_2 by:

$$\begin{aligned} tr_0(s_{\varepsilon \preceq \varepsilon'}) &= s_{\varepsilon \preceq \varepsilon'} \\ tr_1(s_{\varepsilon \preceq \varepsilon'}) &= s_{\varepsilon \preceq \varepsilon'} \\ tr_1(\neg s_{\varepsilon \preceq \varepsilon'}) &= \neg s_{\varepsilon \preceq \varepsilon'} \\ tr_2(s_{\varepsilon \preceq \varepsilon'}, x, y) &= s_{\varepsilon \preceq \varepsilon'} \end{aligned}$$

We can then calculate the translation of $\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ into propositional logic by:

$$F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle} = tr_2 \left(tr_1 \left(nnf(\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}) \right), 0, size \left(tr_1 \left(nnf(\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}) \right) \right) \right)$$

We denote by $Vars(F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle})$ the set of all propositional variables which occur in the formula $F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$.

Theorem 9 *The ES-INF problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan iff the following quantified boolean formula is true:*

$$Q = \bigexists_{s_{\varepsilon \preceq \varepsilon'} \in V_s} \bigwedge_{p \in Vars(F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}) \setminus V_s} p \quad F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$$

SKETCH OF PROOF. (\Rightarrow) Suppose that the ES-INF problem $\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle$ has a solution plan. Then, by Proposition 3, we know that there it has a solution plan $P = \varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ and $\varepsilon_i \neq \varepsilon_j$ for all $i < j$. We consider the valuation v of variables in V_s such that P is the corresponding designated plan (i.e. $\forall i, j$ such that $1 \leq i \leq j \leq k$, we have $v \models s_{\varepsilon_i \preceq \varepsilon_j}$ and $\forall \varepsilon, \varepsilon' \in Op_{\text{inf}} \setminus P$ we have $v \not\models s_{\varepsilon \preceq \varepsilon'}$).

$\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ is valid in the class **M** then, by theorem 2, $tr_1(\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle})$ is valid in the class **K**. By Theorem 3, $F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ is valid.

(\Leftarrow) Let v a valuation of variables in V_s for which Q is true. We can remark that in this case, φ_{V_s} is evaluated to true as it remains as a conjunct of $F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ after application of $tr_2 \circ tr_1$ on $\varphi_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$. We have to prove that the corresponding designated plan $P = \varepsilon_1, \dots, \varepsilon_k$ with $k \leq |Op_{\text{inf}}|$ is a solution plan. We simplify the formula $F_{\langle \Sigma, Op_{\text{inf}}, \alpha_G \rangle}$ with respect to v that leads to a formula equivalent to the translation into propositional logic of $(\bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha) \rightarrow \langle \langle \varepsilon_1 \rangle \rangle \dots \langle \langle \varepsilon_k \rangle \rangle \Box_{\mathbf{m}} \alpha_G$. Then, using respectively Theorem 2, Theorem 3, Proposition 2 and Theorem 6 we can deduce that $\Sigma \models_{\mathbf{M}} \langle \langle \varepsilon_1 \rangle \rangle \dots \langle \langle \varepsilon_k \rangle \rangle \Box_{\mathbf{m}} \alpha_G$ (i.e. P is a solution plan). ■

3.9 Conclusion

In this chapter, we presented an integrated architecture for cognitive planning that considers two core components: the belief revision and the planning module.

We specified two kinds of planning problem: informative and interrogative. A solution plan for an informative planning problem is a sequence of informative actions, while a solution plan for an interrogative planning problem is given by a sequence of informative actions and questions. Moreover, we considered that the questions asked by agent \mathbf{m} to agent \mathbf{h} , are binary, which means there are only two possible answers: yes or no. We defined the default answer to a binary question as the positive answer. Based on the previous assumption, we explored two solutions for an interrogative planning problem, a strong and a weak solution. A strong solution for the interrogative planning problem considers finding a sequence of questions for every possible answer. In contrast, a weak solution for an interrogative planning problem considers discovering a sequence of questions taking into account that the answer is always the default one.

We proved that checking existence of a solution for an informative planning problem is Σ_2^P -complete. Likewise, we proved that checking existence of a weak solution for an interrogative planning problem has the same complexity.

Afterwards, we illustrated the potential for application of our model for cognitive planning by means of two examples. The first one is an instance of an informative planning problem. In this example we considered a HMI scenario in which agent \mathbf{m} is the artificial assistant of the human agent \mathbf{h} . Agent \mathbf{h} has to choose a sport to practice since her doctor recommended her to do a regular physical activity to be in good health. Agent \mathbf{m} 's aim is to help agent \mathbf{h} to make the right choice, given her actual beliefs and desires.

The second example is an instance of an interrogative planning problem in which we shown that our model for cognitive planning can elegantly formalize some principles of the motivational interviewing (MI) methodology, a counseling method used in clinical psychology for eliciting attitude and behavior change in humans.

Finally, in this chapter we introduced an optimal encoding for an informative planning problem using quantified boolean formula (QBF) with an optimal number of quantifiers in the prefix.

In future we plan to extend our analysis to aspects of MI that we were neglected in this chapter, such as: an important strategy of MI consists of helping the participant to overcome the obstacles that prevent her from converting her mere desires into intentions and then into effective behavior. Some of these obstacles are of cognitive nature. For example, the participant could hesitate whether to start to practice a sport regularly since she fears that practicing a sport increases the risk of getting injured. In this situation, the counselor can try to reassure the participant that her fear is unfounded. More generally, it can try to make the participant to revise her beliefs that a certain action has negative consequences. Another cognitive obstacle could be the participant's belief that she does not have the right capabilities and potential to change her behaviour. The counselor can again try to make the participant revise her belief by providing counterevidence.

An Implemented System for Cognitive Planning

In this chapter we present the implementation of example 1 we introduced in Chapter 3 in which an artificial agent has to persuade a human agent to practice a sport based on her preferences¹. The implemented system allows us to represent and reason about other agents' beliefs, desires, and intentions using our logical framework introduced in Chapter 3.

The system has three components: the belief revision, the planning and the translator modules. They work in an integrated way to firstly capture new information about the world, secondly to plan a sequence of speech acts aimed at achieving a persuasive goal and, finally, to verify satisfiability of the formulas generated at each step of the process. We explain how the formulas representing the rules and constraints for a specific problem domain are loaded into the system. The system takes this information as the initial state and some actions — which are of type speech act — to build a plan that leads to the goal. An important feature is that actions have preconditions that impose constraints on their execution order.

The initial GUI of our system was developed using a chatbot interface. Afterwards, we replaced this component with a multimodal web interface developed in cooperation with the IA Enterprise DAVI². The new GUI goes beyond verbal behavior to enhance our artificial agent with the skill to express emotions and gestures.

We conducted experiments to test and compare our artificial agent performance using the brute force technique versus the QBF approach for cognitive planning detailed in Chapter 3.

4.1 Implementation

The functionality of our integrated system for cognitive planning is defined by the use case presented in Figure 4.1.

¹https://github.com/iritlab/artificial_agent

²<https://davi.ai/en/home/>

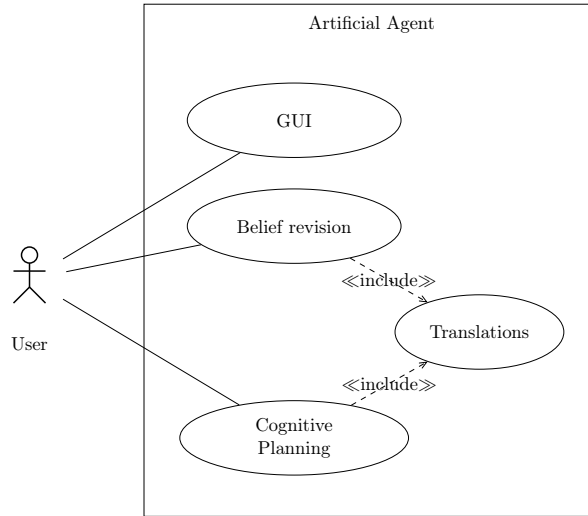


Figure 4.1: Use Case Artificial Agent.

Its two core modules are belief revision and cognitive planning which work in an integrated way. Firstly, the belief revision module reads the input coming from the human through the graphical user interface (GUI) for dialog and verifies that this input does not contradict the core beliefs stored in the belief base. If the input contradicts the core beliefs, then the input is rejected and the belief base is not updated. On the contrary, if the input does not contradict the core beliefs, then the belief base is revised using a maximal consistent subset (MCS) approach whereby the input has priority over the old volatile beliefs.

Secondly, the planning module reads the initial state, the set of actions, and the goal and starts to generate candidate plans of different size, starting with size equal to one. During this phase, the planning module calls the translator module which converts the $\mathcal{L}_{\text{Frag}}^+$ planning formula into its equivalent in propositional logic, following the sequence of reductions detailed in Figure 2.1. After the reduction process performed by the translator, the planning module executes the SAT encoding tool TouIST [Fernandez *et al.* 2020] to verify the validity of the propositional formula. TouIST will encode the formula in CNF format and send it to MiniSAT (this solver is set by default in the application) for checking satisfiability. TouIST can work with external solvers that accept standardized DIMACS as input language. In Figure 4.2 we show the flow diagram for the cognitive planning process:

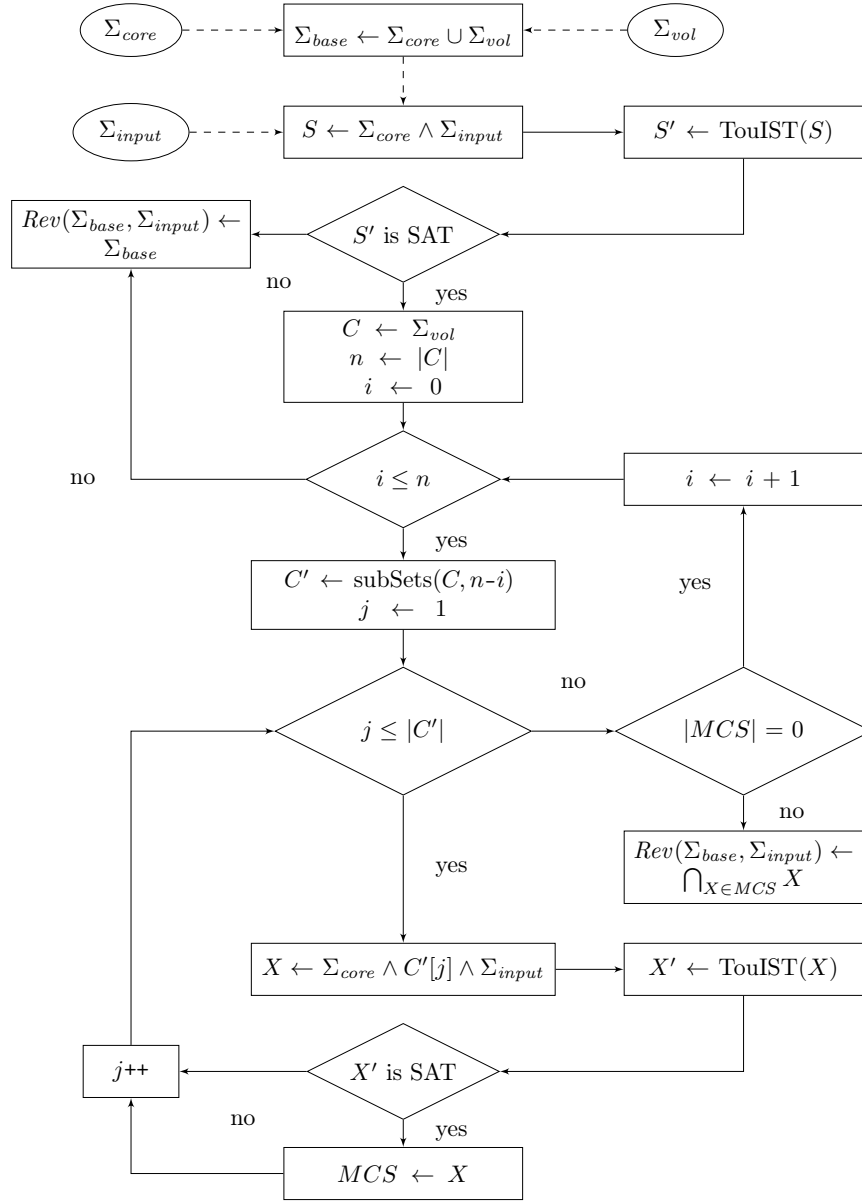


Figure 4.2: Belief revision process flow diagram

In Algorithm 1, $plan_{\mathcal{L}_{Frag}^+}[k, i]$ (line 5) is the i -th candidate plan of size k , generated from the following elements: the belief base Σ_{base} , the i -th element in the set of combinations C of size k from Op with its corresponding pre-conditions and the goal α_G .

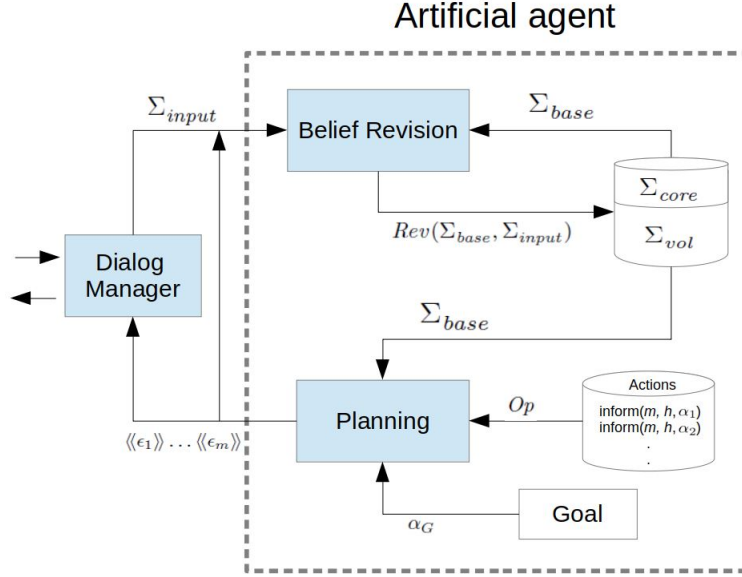


Figure 4.3: System architecture

Algorithm 1 Cognitive planning

```

1: function GENERATEPLANS( $k$ )
2:    $C \leftarrow \text{GETSUBSETS}(Op, k)$ 
3:   for  $i \leftarrow 1$  to  $|C|$  do
4:      $\epsilon \leftarrow \text{PRECONDITIONS}(C[i])$  ▷ Assigns the pre-condition to
each element in  $C[i]$ 
5:      $plan_{\mathcal{L}_{\text{Frag}}^+}[k, i] \leftarrow \neg (\Box_m \Sigma_{base} \Rightarrow \epsilon[i] \Box_m \alpha_G)$ 
6:      $plan_{\mathcal{L}_{\text{Prop}}}[k, i] \leftarrow \text{TRANSLATOR}(plan_{\mathcal{L}_{\text{Frag}}^+}[k, i])$  ▷ Call Translator module
7:     if  $\text{TOUIST}(plan_{\mathcal{L}_{\text{Prop}}}[k, i]) = \text{UnSAT}$  then ▷ Call TouIST solver
8:       Print “Plan  $i$  of size  $k$  is valid”
9:       return 0 ▷ Exit
10:    end if
11:  end for
12: end function
13:
14:  $k=1$ 
15: while ( $k \leq |Op|$ ) do ▷  $Op$  is the set of actions
16:   GENERATEPLANS( $k$ ) ▷  $k$  is the size of the plan
17:    $k=k+1$ 
18: end while
19: Print “Plan not found”

```

In Figure 4.3 we show the detailed system architecture based on the general one sketched in Section 3.2.

In order to probe the potential of our framework for cognitive planning we implemented the example presented in Section 3.6. In this scenario agent m is the artificial assistant of the human agent h . Agent h has to choose a sport to practice since her doctor recommended her to do a regular physical activity to be in good health.

Agent m 's aim is to help agent h to make the right choice, given her actual beliefs and desires. Consequently, to set the initial belief base agent m has to be provided with information about the possible options that the user can choose (Opt) and their properties (Var). For each pair (Opt, Var) we have a valuation Val as is shown in Table 3.1

Formulas representing the rules and constraints are loaded as part of agent m 's belief base. For example, the implementation of the formula representing the fact that agent h explicitly believes that a sport cannot have two different values for a given property is formalized as follows:

$$\bigwedge_{\substack{o \in Opt \\ x \in Var \\ v_1, v_2 \in Val_x: v_1 \neq v_2}} (\Delta_h \text{val}(o, x \mapsto v_1) \rightarrow \Delta_h \neg \text{val}(o, x \mapsto v_2))$$

The syntax for writing the formulas is based on the TouIST language, with the extension of the modal operators for explicit and implicit belief. For example, we use $\{h\}$ for representing Δ_h . Similarly we use $[m]$ for \square_m .

```
bigand $o, $x, $v1, $v2 in $Opt, $Var, $Val($x), $Val($x) when $v1 != $v2:
  {h}val($o, ass($x, $v1)) => {h}not val($o, ass($x, $v2))
end
```

Thus, this syntax allows us to represent functions like the one included in the next formula, which states that an option o is ideal for agent h if and only if the option satisfies all agent h 's desires:

$$\bigwedge_{o \in Opt} (\text{ideal}(h, o) \leftrightarrow \bigvee_{\Gamma \in 2^{Des}} (\text{des}(h, \Gamma) \wedge \bigwedge_{\gamma \in \Gamma} f_{comp}(o, \gamma)))$$

The function f_{comp} specifies, for every option $o \in Opt$ and possible desire $\gamma \in Des$, the condition guaranteeing that o satisfies (or, complies with) γ :

$$\begin{aligned} f_{comp}(o, a) &= \text{val}(o, a), \\ f_{comp}(o, \sim a) &= \neg \text{val}(o, a), \\ f_{comp}(o, [d_1, \dots, d_k] \rightsquigarrow d) &= \neg f_{comp}(o, d_1) \vee \dots \vee \\ &\quad \neg f_{comp}(o, d_k) \vee f_{comp}(o, d). \end{aligned}$$

The full implementation of the formula with the function f_{comp} included, requires the capture of the human's desires which together with the set of rules and constraints are used to generate the machine's belief base.

```

1
2 $n1 =2
3 $n2 =1
4 $n3 =1
5
6 $Delta0      = [
7 "ass(env,land)",
8 "ass(intens,med)",
9 "not ass(loc,indoor)",
10 "ass(cost,high) => ass(soc,mixed)"
11 ]
12
13 $Delta0_1(1)  = ["ass(env,land)"]
14 $Delta0_1(2)  = ["ass(intens,med)"]
15 $Delta0_2(1)  = ["not ass(loc,indoor)"]
16 $Delta0_2(1,1) = ["ass(loc,indoor)"]
17 $Delta0_3(1)  = ["ass(cost,high) =>
18                 ass(soc,mixed)"]
19 $Delta0_3(1,1) = ["ass(cost,high)"]
20 $Delta0_3(1,2) = ["ass(soc,mixed)"]
21
22 $Opt = [sw, ru, te, hr, so, yo, di, sq]
23 $Var = [env, loc, soc, cost, danger, intens]
24
25 $Val(env)      = [land,water]
26 $Val(loc)      = [indoor, outdoor, mixed]
27 $Val(soc)      = [single, team, mixed]
28 $Val(cost)     = [low, med, high]
29 $Val(danger)   = [low, med, high]
30 $Val(intens)   = [low, med, high]
31 .
32 .
33 .
34 bigand
35   $o in $Opt :
36   ideal(h,$o) <=>
37     ((bigand $d0, $i, $e
38       in $Delta0, [1..$n1], $Delta0_1($i)
39       when $d0 in $Delta0_1($i):
40         val($o,$e)
41     end) and
42     (bigand $d0, $i, $e
43       in $Delta0, [1..$n2], $Delta0_2($i,1)
44       when $d0 in $Delta0_2($i):
45         not val($o,$e)
46     end) and
47     (bigand $d0, $i, $p, $c
48       in $Delta0, [1..$n3], $Delta0_3($i,1), $Delta0_3($i,2)
49       when $d0 in $Delta0_3($i):
50         not val($o,$p) or val($o,$c)
51     end))
52 end

```

Listing 4.1: Formulas representing the model of the human agent's mind

The set of human desires is represented by $\$Delta0$ in the previous syntax. The counters $\$n1, \$n2, \$n3$ specify the number of positive desires, negative desires and conditional desires respectively. We conceive a positive desire as the human expressing a valuation for a variable assignment (e.g., environment is land). A negative desire is a negative valuation for a variable assignment. Finally, a conditional

desire is a conditional valuation between variable assignments (e.g., if the cost is high then sociality level should be mixed).

Similarly, the goal to be achieved by the planning agent is captured by the following formula:

$$\alpha_G \stackrel{\text{def}}{=} \bigvee_{o \in Opt} \text{potIntend}(\mathfrak{h}, o).$$

Moreover, we suppose that, for agent \mathfrak{h} to have a potential intention to choose option o , denoted by $\text{potIntend}(\mathfrak{h}, o)$, she must have a justified belief that o is an ideal option for her:

$$\text{potIntend}(\mathfrak{h}, o) \stackrel{\text{def}}{=} \Delta_{\mathfrak{h}} \text{ideal}(\mathfrak{h}, o) \wedge \text{justif}(\mathfrak{h}, o).$$

The latter is defined using the same syntax and in our case is expressed by the next formula:

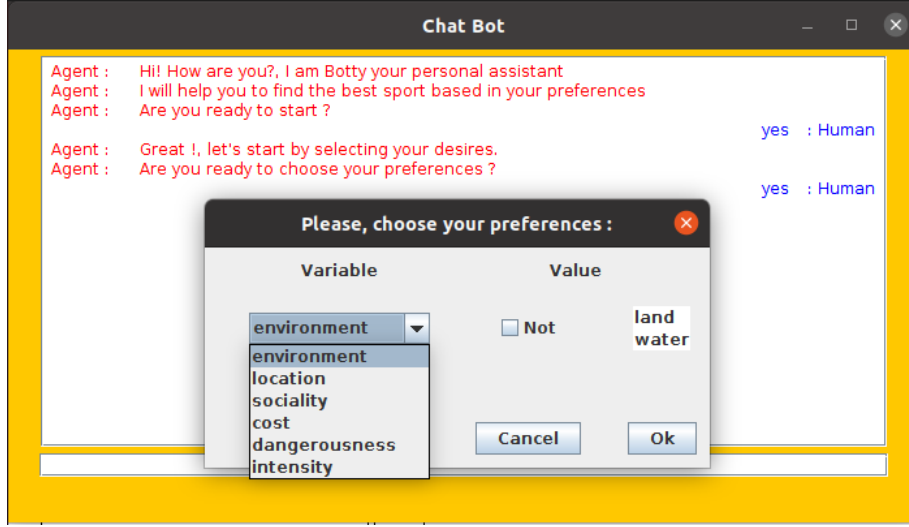
```

1 $Opt = [sw, ru, te, hr, so, yo, di, sq]
2 .
3 .
4 bigor $o in $Opt :
5     {h}ideal(h,$o) and justif(h,$o)
6 end
7 .
8 .

```

Listing 4.2: Formula representing α_G

The set of actions are generated from Table 3.1. For instance, `convince(m,h,val_so_ass_env_land)` is an informative action. It is interpreted as the speech act used by agent \mathfrak{m} to inform agent \mathfrak{h} that the valuation of the property:environment for the option:soccer is land. In order to help agent \mathfrak{h} to select an activity, agent \mathfrak{m} also needs information about \mathfrak{h} 's desires. This information is gathered by agent \mathfrak{m} during its interaction with agent \mathfrak{h} . The interactive interface between \mathfrak{h} and \mathfrak{m} is shown in Figure 4.4. The belief revision module is called after each agent \mathfrak{h} 's feedback and it restores consistency of the agent \mathfrak{m} 's belief base, in case the incoming information is inconsistent with agent \mathfrak{m} 's pre-existent beliefs.

Figure 4.4: Collecting agent h 's preferences

In the example, agent h would like to practice a land activity, with medium intensity, which is not exclusively indoor, and which can be practiced both in single and team mode, if its cost is high. The next rule for precondition states that agent h must be informed by agent m about the dangerousness level of a sport, before presenting other properties for an option. For $a \notin Assign_{dan}$:

$$\mathcal{P}\left(\text{convince}(m, h, \text{val}(o, a))\right) = \square_m \left(\text{val}(o, a) \wedge \bigwedge_{v \in Val_{dan}} (\text{val}(o, \mathbf{dan} \mapsto v) \rightarrow \Delta_h \text{val}(o, \mathbf{dan} \mapsto v)) \right)$$

Next we illustrate how the precondition is assigned (lines 9 to 14) by the planning module together with its $+_m\alpha$ operator in order to specify the successful occurrence of an informative action:

```

1 (not ([m] (
2 (val_di_ass_cost_high => not val_di_ass_cost_low) and
3 (val_di_ass_cost_high => not val_di_ass_cost_med) and
4 (val_di_ass_cost_low => not val_di_ass_cost_high) and
5 .
6 .
7 .
8 )=>
9 ([m](val_te_ass_danger_med and plusa(th_val_te_ass_danger_med,
10 ([m](val_te_ass_intens_med and (val_te_ass_danger_med =>
    th_val_te_ass_danger_med )) and plusa(th_val_te_ass_intens_med,
11 ([m](val_te_ass_soc_mixed and (val_te_ass_danger_med =>
    th_val_te_ass_danger_med )) and plusa(th_val_te_ass_soc_mixed,
12 ([m](val_te_ass_loc_mixed and (val_te_ass_danger_med =>
    th_val_te_ass_danger_med )) and plusa(th_val_te_ass_loc_mixed,
13 ([m](val_te_ass_env_land and (val_te_ass_danger_med =>
    th_val_te_ass_danger_med )) and plusa(th_val_te_ass_env_land,
14 ([m](ideal_h_te and justif_h_te and plusa(th_ideal_h_te,
15 [m]
16 ((th_ideal_h_di and justif_h_di) or (th_ideal_h_hr and justif_h_hr) or (
    th_ideal_h_ru and justif_h_ru) or (th_ideal_h_so and justif_h_so) or (

```

```

th_ideal_h_sq and justif_h_sq) or (th_ideal_h_sw and justif_h_sw) or (
th_ideal_h_te and justif_h_te) or (th_ideal_h_yo and justif_h_yo))
17 ))
18 )

```

Listing 4.3: Preconditions assigned to each action by the system during the planning process

The planning module generates plans with the elements contained in the action file. It starts with plans of length 1, and enters in a loop. At each interaction the planning module asks the SAT solver to verify whether the plan allows to achieve the goal. If no plan of length k is found, the program will increase the counter in one and look for a plan of length $k + 1$. An example of an abstract plan generated by the planning module is:

```

plus({h})(val_te_ass_danger_med)
plus({h})(val_te_ass_intens_med)
plus({h})(val_te_ass_soc_mixed)
plus({h})(val_te_ass_loc_mixed)
plus({h})(val_te_ass_env_land)
plus({h})(ideal_h_te)

```

The order of speech acts is determined by the preconditions. Specifically, the planning module informs firstly about the dangerousness level of the sport. Secondly, it provides explanation of why the user's desires are satisfied. Finally, it indicates the ideal sport for the user, in this case tennis.

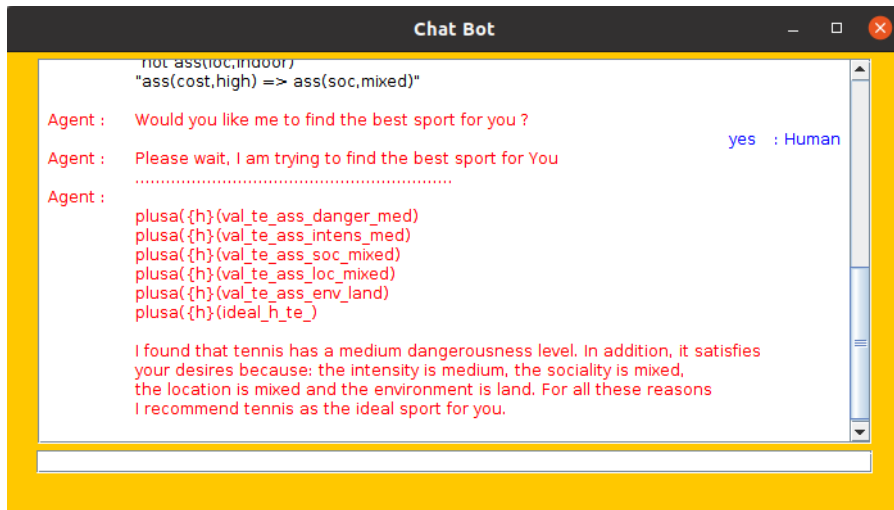


Figure 4.5: Plan shown by the chatbot to the human

The chatbot writes both the sequence of speech acts and its translation into natural language expressions. We decided to display the abstract plan in the GUI, as shown in Figure 4.5, for illustrative purposes oriented to demonstrate how the GUI transforms it into natural language using a simple function. The abstract plan will not be displayed by the GUI in the end-user version of the system.

We improved our system GUI module by replacing the chatbot interface with a web avatar³ developed by the IA Enterprise DAVI as part of joint work on the ANR project: Cognitive Planning in Persuasive Multimodal Communication (CoPains)⁴. The avatar allows us to endow the artificial agent with emotions and gestures, as is shown in Figures 4.6 and 4.7.

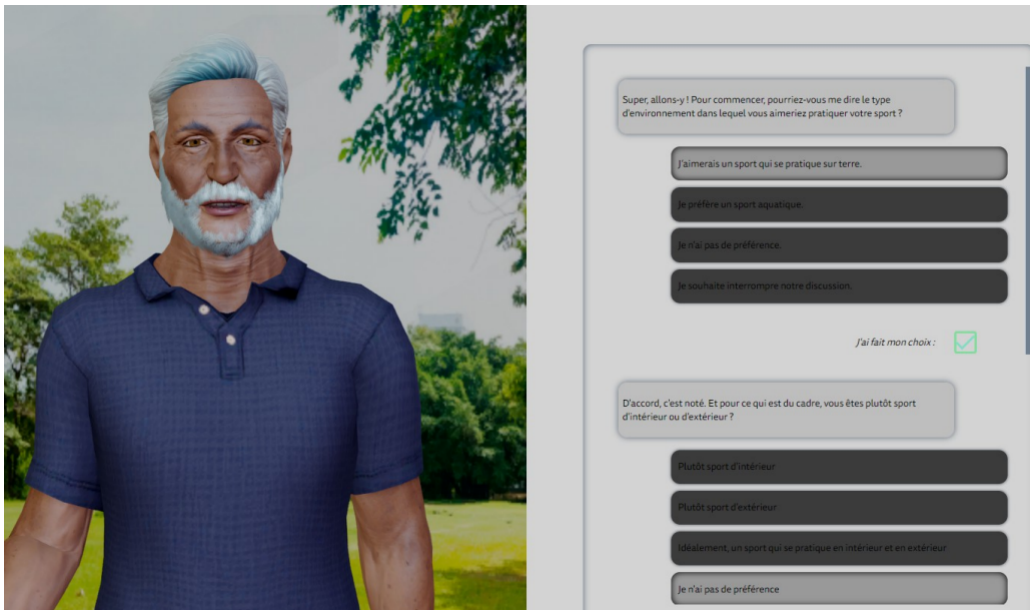


Figure 4.6: Avatar developed in cooperation between IRIT and DAVI

The avatar interacts with the human user to collect her preferences:

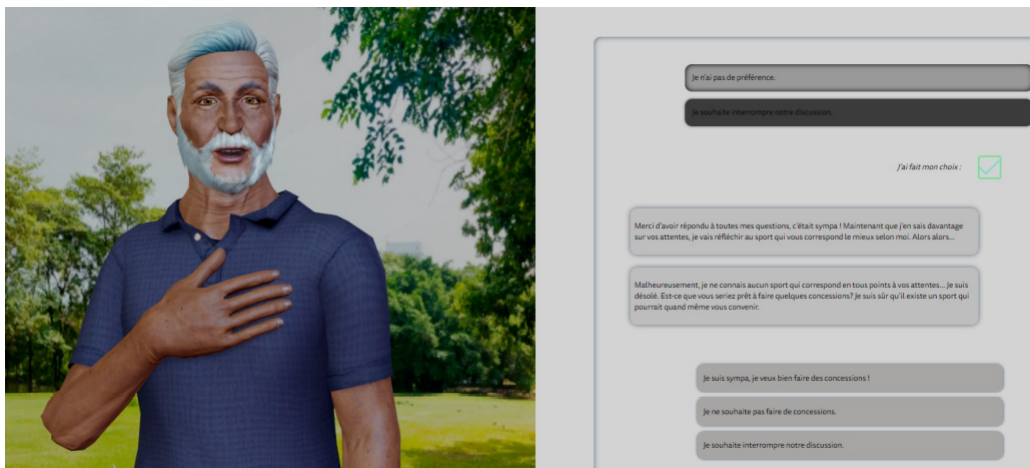


Figure 4.7: Avatar developed in cooperation between IRIT and DAVI

³<https://cognitive-planning.schm.fr/>

⁴<https://www.irit.fr/CoPains/>

After the avatar performs the planning process, it recommends the ideal sport to the human (Figure 4.8):

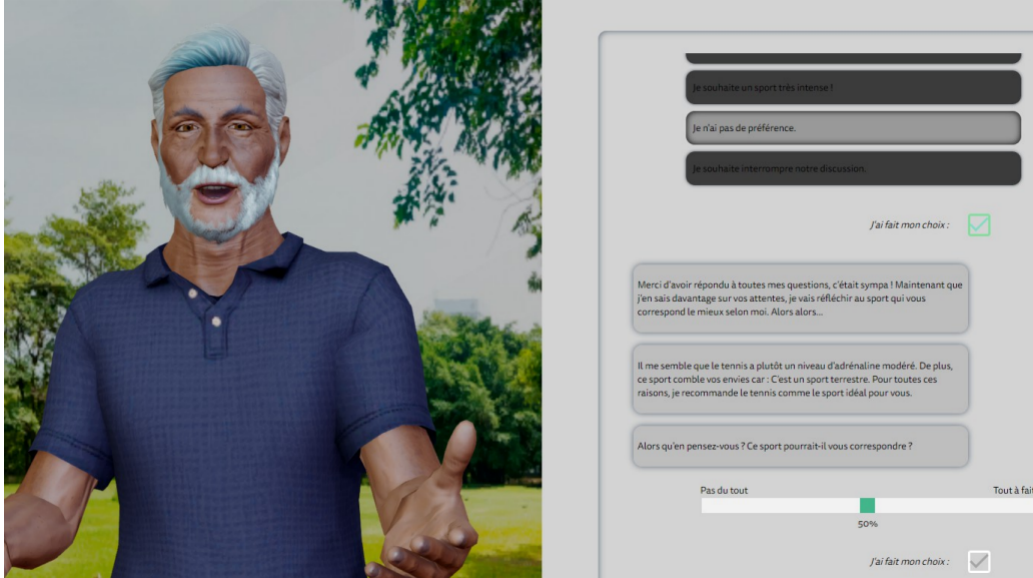


Figure 4.8: Avatar developed in cooperation between IRIT and DAVI

The belief revision and planning modules were encapsulated in a web api. The GUI was installed on a different server in order to improve the system performance.

4.2 Experiments

In this section, we present the experiments conducted in order to test the implemented cognitive planning system. The experiment was devoted to evaluate the performance of the planning module in integration with the belief revision module. The GUI was not used during the test, therefore the procedure was carried out on command line mode.

In order to perform the test we generate first a set of desires of the human in different input files. These input files are processed by the belief revision module sequentially in order to generate the volatile side of the belief base. Second, the translator module is called to generate the initial state and the goal. Finally, the initial state, the set of actions (repertoire of speech acts) and the goal are used to call the planning module.

The set of options and variables described in Table 3.1 were used to test the performance of the system, expanding the table in the number of sports available. Similarly, we vary the number of the human's desires. Figure 4.9 shows the results of the computation. The data plotted in the previous graph are shown in Table 4.1.

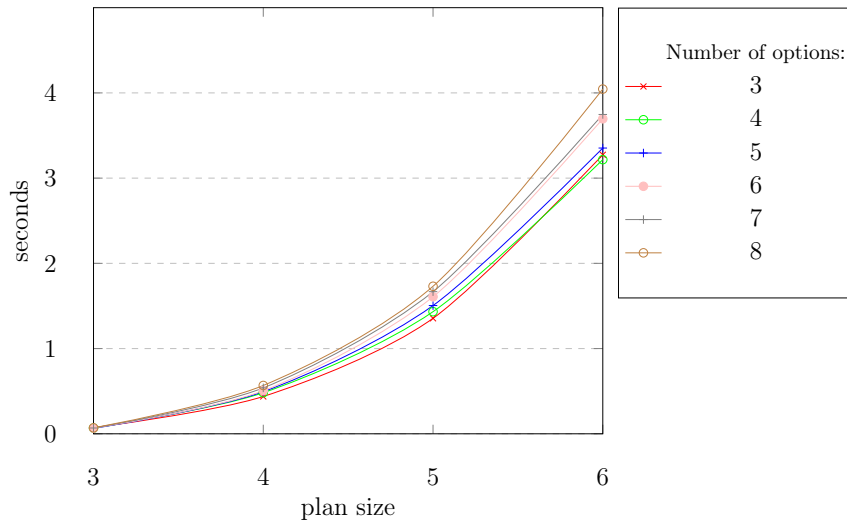


Figure 4.9: Processing time consuming by the brute force approach based on the number of options.

Plan size	Number of options (sports)					
	3	4	5	6	7	8
3	0.059	0.067	0.063	0.066	0.068	0.070
4	0.438	0.482	0.494	0.506	0.539	0.567
5	1.355	1.433	1.505	1.608	1.668	1.731
6	3.274	3.217	3.353	3.696	3.747	4.045

Table 4.1: Processing time (in seconds) to achieve a plan based of the number of options

We now present the experiments conducted in order to compare the brute force and the QBF approach, applied to our implemented system for cognitive planning. The experiment was devoted to evaluate the performance of the planning module based on the time needed to find a valid plan. We also evaluate the performance of the translator module during the transformation process of the planning formula into its equivalent in propositional logic or quantified boolean formula. Finally, we compare the execution times required by the SAT and QBF solver to solve the formulas.

In Table 4.2 we present the data corresponding to the time in seconds used by the planning module in order to find a valid plan:

Plan size	Planning module processing time (seconds)	
	Brute force	QBF
4	0.464	10.015
5	2.059	10.936
6	12.610	12.343
7	61.920	12.837

Table 4.2: Planning module processing time required to achieve a plan.

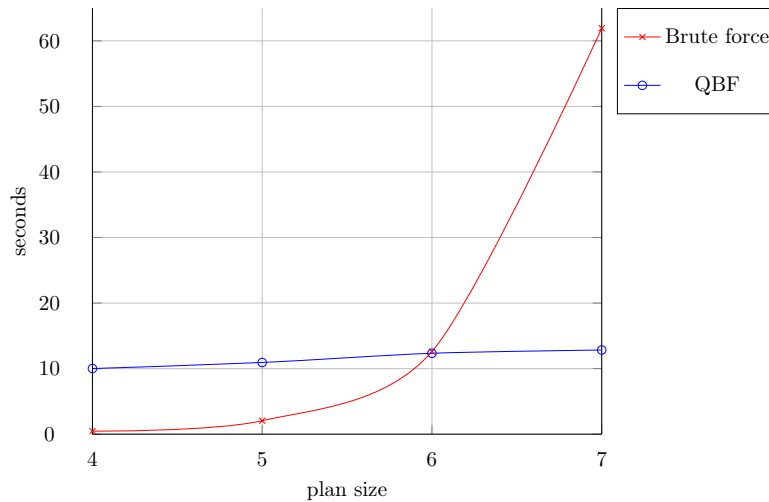


Figure 4.10: Comparison in performance between the brute force and the QBF approaches.

To test the performance of the translator module in applying the set of transformations shown in Figure 2.1, we carried out an independent analysis of this module. The following table shows the difference in time when transforming the planning formula to their equivalent propositional logic or QBF formula (column *a*). Similarly, we evaluate the time used by TouIST in order to solve the propositional logic formula using MiniSat⁵ as well as the time used to solve the QBF formula using RAReQS⁶ (column *b*).

⁵<http://minisat.se/>

⁶<http://sat.inesc-id.pt/~mikolas/sw/areqs/>

Plan size	Time in seconds used by the Translator module (a)		Time in seconds used by TouIST (b)	
	Propositional Logic	QBF	MiniSat	RAReQS
4	0.009	1.840	0.013	8.240
5	0.009	2.013	0.014	8.539
6	0.019	2.233	0.026	10.038
7	0.013	2.520	0.027	10.402

Table 4.3: Translator module and TouIST solver processing times

These experiments were conducted using an Ubuntu 64 bits linux virtual machine running on a core i7 processor with 16 gigabytes RAM.

4.3 Discussion

The architecture presented in Figure 4.3 works as an integrated system. All the processes, interfaces and exchange of data between the modules are working according to the definition of the use cases displayed in Figure 4.1.

The system dialogue capability is limited for the moment. The human agent communicates her desires to the machine, and the latter computes the most suitable plan. There is no feedback from the human after the sequence of speech acts performed by the machine.

We observed that the brute force approach is efficient when the size of the plan is small (not more that 5 actions in the length of the plan), but when the size increases the computation tends to be exponential. In the search for an alternative to the brute force algorithm, we implement and test the optimal QBF encoding detailed in Section 3.8.

However, it would be possible to include a heuristic to improve the performance of the general process. For example, the planning module could consider the size of the input (based on the human's set of desires) as the initial size of the plan. Thus, the planner will generate plans of that size at least. This prevents the planner from spending time to generate candidate plans of smaller size than the number of human's desires. In addition, an optimization can be included in the algorithm if we add a mechanism for giving priorities to certain types of actions. For example, the actions which are stated as preconditions should be prioritized to be included between the first sets of combinations to be tested by the planning module.

In the brute force approach, we need to call the SAT solver to verify satisfiability for each candidate plan generated by the system until a valid plan is achieved. In contrast, with the QBF approach, the system generates a single formula that considers all the possible plans. Moreover, with the QBF approach, the time needed for solving the cognitive planning problem has a stable average time that does not depend on the plan size (Table 4.2). This performance improvement can be explained by the fact that the QBF solver parallelizes the solving process by creating multiple solver instances.

We found that the time needed by the translator module when it transforms the planning formula with optimal QBF encoding into its equivalent QBF formula is more significant (nearly a 200:1 ratio) than the time needed to transform the SAT planning formula into its propositional logic version (Table 4.3 column a). This difference can be explained by the additional process the translator module needs to perform in order to generate the existential and universal variables for the QBF formula.

Similarly, we found that the time consumed by the RAReQS solver to solve the QBF formula is also more significant than the time needed by the MiniSAT solver to solve the propositional logic formula. In Table 4.3 column (b), we can see that this resolution time proportion varies between 600:1 and 300:1, depending on the size plan.

However, this important difference in performance between the brute force and the QBF approach, is overcome by the QBF encoding because in the brute force approach it is necessary to call the SAT solver for each candidate plan until a valid plan is found, while in the QBF approach we only need to call the QBF solver once.

4.4 Conclusion

In this chapter, we described the steps performed during the implementation of example 1 described in Chapter 3. We found that the general architecture proposed in Section 3.2 provides the right separation between functionalities and allows us to organize the application code in two layers front-end (interface module) and back-end (belief revision and planning modules). We encapsulated the set of transformations shown in Figure 2.1 in a separate component called translator module. We verified that all the components interact between them as an integrated system.

We found that functional programming was the correct choice for implementing the back-end modules, in particular the translator module, thanks to its well-known feature of effectively dealing with recursive calls. Another important feature of the language that we want to remark is that it allows to capture the abstract syntax tree (AST) during the translation process. This last property allowed us to track the translation of formulas expressed in our NP-fragment at each step of the transformation process.

Moreover, our system's modular design allowed us to detach our initial chatbot interface and replace it with a web interface developed in cooperation with a commercial company dedicated to building artificial agents endowed with emotions and gestures.

We performed experiments to compare our two approaches for finding a solution for informative planning problems, the brute force technique based on SAT and the QBF approach. The experiments demonstrated that the QBF approach outperforms the brute force method as plan size increases.

Our main contribution is that we have built our system in a modular way by designing the different components, including planning and belief revision, with the

necessary interfaces to work in an integrated way. This feature allows the scalability of the system.

The results of this chapter lead to many interesting directions for future work. We aim to implement the interrogative planning problem of the scenario described in Section 3.7 and evaluate the performance of the artificial agent in its interaction with the human. Moreover, we could combine our implementation of cognitive planning with machine learning and data mining techniques, as presented in [Krzywicki *et al.* 2016], in order to extract information about the human user from real data. We also could use a learning function to let the system select the most convenient approach, brute force or QBF, depending on the plan size. We plan to include a security module for granting access to the system in a multi-user environment.

A Logical Modeling of the Yōkai Board Game

In Chapter 4 we implemented an example of an informative planning problem with a chatbot application that includes the core components of our architecture sketched in Figure 4.3. However, the chatbot implementation does not consider three aspects we are interested in evaluating. First, the logical framework used in the chatbot implementation does not incorporate the temporal aspect. Therefore, the artificial agent is not able to reason about other agents' past or future beliefs. Second, the chatbot implementation explores only one direction of the interaction between the two agents. Accordingly, the belief revision module revises the belief base taking into account the feedback from only one participant (the human agent). Finally, the chatbot application uses a single goal for planning. We want to explore the option of considering a set of goals, among which the artificial agent will select the best one following a strategy.

In this chapter we apply our model of cognitive planning to the case of the Yōkai cooperative board game¹. Yōkai requires a combination of Theory of Mind (ToM), temporal and spatial reasoning to be played effectively by an artificial agent. We extend our logical framework to properly account for these three dimensions. Furthermore, we show that the belief revision module can revise the belief base with information related to the actions performed by the human and the artificial player. Moreover, we load a set of goals into the system that allow the artificial player to select the best strategy to win the game in cooperation with the human player.

Finally, we provide experimental results to compare two game configurations: human-machine and human-human collaboration.

5.1 Introduction

When one wishes to model socio-cognitive agents and, in particular, agents endowed with a Theory of Mind (ToM) who are capable of reasoning about other agents' beliefs, some of the privileged tools are epistemic logic (EL) [Fagin *et al.* 1995,

¹<https://github.com/iritlab/yokai>.

[Halpern & Moses 1992] and its extensions by informative and communicative extensions such as public and private announcements [Gerbrandy & Groeneveld 1997, Plaza 1989, Baltag *et al.* 1998]. The latter belongs to the Dynamic Epistemic Logic (DEL) family [van Ditmarsch *et al.* 2007].

The major disadvantage of EL and DEL is that they have most of the time a high complexity thereby making them not very well-suited for practical applications. In particular, extending multi-agent EL by simple notions of state eliminating public announcement or arrow eliminating private announcement does not increase its PSPACE complexity [Lutz 2006, Bolander *et al.* 2015b]. However, the satisfiability problem of full DEL with public, semi-private and private communicative actions was shown to be NEXPTIME-complete [Aucher & Schwarzenrüber 2013]. The situation is even worse in the context of epistemic planning: it is known that epistemic planning in public announcement logic (PAL) is decidable, while it becomes undecidable in full DEL, due to the fact that the epistemic model may grow as a consequence of a private announcement [Bolander & Andersen 2011].

In [Lorini 2020, Lorini 2018], a variant of epistemic logic with a semantics exploiting belief bases is introduced. It distinguishes explicit belief from implicit belief. The former is modeled as a fact in an agent’s belief base, while the latter is modeled as a fact that is deducible from the agent’s explicit beliefs. The main advantages of the belief base semantics for epistemic logic compared to the standard possible world semantics based on multi-relational structures (so-called Kripke models) are (i) its compactness, and (ii) its closeness to the way artificial cognitively-inspired agents are traditionally modeled in AI and in the area of knowledge representation and reasoning (KR) by adopting a database perspective [Shoham 2009]. In [Lorini & Romero 2019], it is shown that this variant of epistemic logic provides a valuable abstraction for modeling multi-robot scenarios in which each robot is endowed with a ToM whereby being able to ascribe epistemic states to the other robots and to reason about them.²

In this chapter, we leverage the belief base semantics for epistemic logic to model interaction in the context of the cooperative board-game Yōkai.³ We consider its two-player variant in which an artificial agent has to collaborate with a human agent to win it and to obtain the best score as possible. Yōkai is an interesting testbed for artificial agents, as it covers a lot of epistemic and strategic reasoning aspects as well as planning and belief revision aspects. The idea of testing the performance of artificial agents in the context of cooperative board-games in which ToM reasoning plays a role is not new. Some works exist about modeling and implementing artificial players for the card game Hanabi [Bard *et al.* 2020, Eger *et al.* 2017, Eger & Martens 2017]. Yōkai adds to the ToM dimension, which is central in Hanabi, the temporal and spatial dimension. First of all, in Yōkai a player’s performance relies on her/its capacity to remember the cards she/it and the other player have seen in the past. Secondly, the players must move cards in a shared space and there are spatial restrictions on

²See also [Bolander 2014, Dissing & Bolander 2020] for a DEL-based approach to modeling and implementing ToM on social robots.

³<https://boardgamegeek.com/boardgame/269146/ykai>

card movements that should be taken into account by the players. More generally, the interesting feature of Yōkai, from the point view of KR, is the combination of epistemic, temporal and spatial reasoning that is required to completely apprehend all the game facets and dimensions.

The main novelty of our approach to modeling artificial board-game players is the use of SAT-based techniques. Specifically, the language we present for representing the artificial player’s beliefs about the static and dynamic aspects of the game as well as about the human player’s beliefs has the same complexity as SAT and can be polynomially translated into a propositional logic language. This opens up the possibility of exploiting SAT techniques for automating reasoning of the artificial player in the context of the Yōkai board-game.

The chapter is organized as follows. In Section 5.2, we explain the rules of Yōkai and clarify the representation and reasoning requirements that are necessary for the artificial player to be able to play the game in a clever way. In Section 5.3, we introduce the specification language for modeling the artificial player’s actions and beliefs about the game properties and about the human player’s beliefs. It is a timed language for explicit and implicit belief with a semantics exploiting belief bases. The main novelty compared to the epistemic language presented in [Lorini 2020] is the temporal component: the artificial player modeled in the language has knowledge about the current time of the game and beliefs about current and past beliefs of the human player. Thereafter, in Section 5.4 the artificial agent architecture is explained. We provide details regarding the two main components of the system: the action selection and the belief change. Finally, Section 5.5 is devoted to the encoding of the game in the language. We first model the static aspects, namely, the artificial agent’s beliefs about the rules and properties of the game. Then, we focus on the dynamic aspects by modeling the artificial player’s actions and how they are used for planning a sequence of moves at a given round of the game.

5.2 Game description

In this section, we explain the rule of the Yōkai game. As pointed out in the introduction, we only consider the two-player variant of the game with an artificial and a human player.

Colored cards and actions. There are 4 types of card, red, yellow, green and blue cards, and 4 cards for each color. This gives us a total amount of 16 cards. The cards are placed face down in a grid of size 4×4 in a random way, so that the players cannot see their colors. The goal of the game is to gather together the cards of the same color, as illustrated in Figure 5.1a.

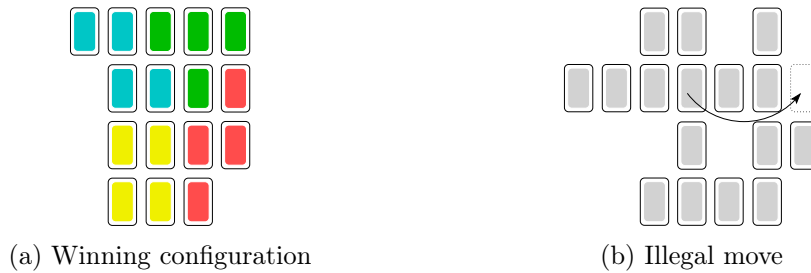


Figure 5.1: Examples of game configuration

The players play sequentially in each round of the game. At each round, each player has to play the following 4 actions in the order she/it prefers:

1. to look at two cards privately (2 actions),
2. to move one card from its current position to a new position adjacent to another card (that is, linked to the latter by one of its sides) and without separating the cards into two disjointed groups,
3. either to activate a hint from the set of available hints (see next paragraph) or to disclose an information by marking one card with an active hint.

Figure 5.1b portrays an example of illegal move. Indeed, the move will separate the cards into two independent groups.

Hints. Three types of hint are available: 1-color, 2-color and 3-color hints. Specifically, 1-color, 2-color and 3-color hints indicate, respectively, a single color, two colors and three colors in the set {red, green, blue, yellow}. There are 14 different hints available, but only seven hints are randomly selected at the beginning of the game: two occurrences of 1-color hint, three occurrences of 2-color hint and two occurrences of 3-color hint.

A hint can be used after being activated, and only available hints can be activated. Moreover, it can be used only once throughout the game. The state of a hint is unique: it is either available, activated, or marking. In particular, after a hint has been activated and used to mark a card, it is no longer available or activated. For example, suppose a player just looked at one card and discovers it is a red card. She can mark it with an active 2-color hint of type red/green to inform the other player that it is either a red card or a green card.

A hint is properly used if one of its colors matches with the color of the card it marks. For example, if a player uses an active 2-color card of type green/blue to mark a blue card, then the hint is properly used. Conversely, if the player uses it to mark a red card, then the hint is improperly used. Note that unless a mistake has been made, hints must be used properly.

Finally, when an active hint marks a card, that card can no longer be moved or observed.

End of the game. When a player thinks that the goal state is achieved, she/it can decide to stop the game. Otherwise, the game ends when all hints have been used by the players. At the end of the game, the cards are turned face up. If the goal state is achieved, the players win the game. Otherwise, they lose it. In case of win, the final score is calculated as follows: $score = x_1 - x_2 + 2x_3 + 5x_4$ where x_1 is the number of properly used hints, x_2 is the number of improperly used hints, x_3 is the number of activated hints that were not used by the players, and x_4 is the number of non-activated hints. The final score ranges in the interval $[-7, 35]$.

Requirements. As emphasized in the introduction, in order to be able to reason about the static and dynamic aspects of the game, a player must have beliefs about:

- the other player’s actual beliefs (*ToM reasoning*);
- the current positions of the cards and the executable card movements, given the current spatial configuration of the game (*spatial reasoning*);
- the color of the cards she/it observed in the past as well as the other player’s past observations (*temporal reasoning*).

5.3 A timed language for explicit and implicit belief

This section presents a two-agent timed variant of the language and the semantics of the logic of explicit and implicit belief presented in [Lorini 2020]. The two agents are the artificial agent (or machine) \mathbf{m} and the human user \mathbf{h} . Agents \mathbf{m} and \mathbf{h} are treated asymmetrically. Our language allows us to represent (i) \mathbf{h} ’s explicit beliefs at different points in a game sequence, and (ii) \mathbf{m} ’s actual explicit and implicit beliefs, namely, \mathbf{m} ’s explicit and implicit beliefs at the current time point of the game sequence. Following [Lorini 2020], explicit beliefs are defined to be beliefs in an agent’s belief base, while implicit beliefs are those beliefs that are derivable from the agent’s explicit beliefs.

We first present the static language in which agent \mathbf{m} ’s beliefs do not change. Then, we present a dynamic extension in which agent \mathbf{m} ’s belief base can be expanded by new information.

5.3.1 Static language

Assume a countably infinite set of atomic propositions Atm . We define the language in two steps.

We first define the language $\mathcal{L}_0(Atm)$ by the following grammar in BNF:

$$\alpha ::= p^t \mid \Delta_{\mathbf{h}}^t \alpha \mid now^{\geq t} \mid \neg \alpha \mid \alpha_1 \wedge \alpha_2 \mid \Delta_{\mathbf{m}} \alpha$$

where p ranges over Atm . $\mathcal{L}_0(Atm)$ is the language for representing agent \mathbf{h} ’s timed explicit beliefs and agent \mathbf{m} ’s actual explicit beliefs. Specifically, the formula

$\Delta_{\mathfrak{h}}^t \alpha$ is read “agent \mathfrak{h} explicitly believes at time t that α is true”, whilst $\Delta_{\mathfrak{m}}$ is read “agent \mathfrak{m} actually has the explicit belief that α ”. The beliefs of the machine \mathfrak{m} are not indexed by time because our language only allows us to talk about the beliefs of \mathfrak{m} in the present tense. (It is not a question here of a technical impossibility, but the language thus extended would introduce a greater expressiveness of the language not necessary here to model our problem.) Furthermore, it is important to note that past beliefs are accessible from previous belief states (see below the section on revision of the belief base). Atomic propositions are assumed to be timed: p^t is read “atomic proposition p is true at time t ”. Finally, formula $now^{\geq t}$ provides information about the current time point. It is read “the actual time of the game play is at least t ”.

Then, we define $\mathcal{L}_0^T(Atm)$ to be the subset $\mathcal{L}_0(Atm)$ including only timed formulas, that is:

$$\begin{aligned} \mathcal{L}_0^T(Atm) = & \{p^t : p \in Atm \text{ and } t \in \mathbb{N}\} \cup \\ & \{\Delta_{\mathfrak{h}}^t \alpha : \alpha \in \mathcal{L}_0(Atm) \text{ and } t \in \mathbb{N}\} \cup \\ & \{now^{\geq t} : t \in \mathbb{N}\}. \end{aligned}$$

Elements of $\mathcal{L}_0^T(Atm)$ are denoted by x, y, \dots

The language $\mathcal{L}(Atm)$ extends the language $\mathcal{L}_0(Atm)$ by a modal operator of implicit belief for agent \mathfrak{m} and is defined by the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Box_{\mathfrak{m}} \alpha,$$

where α ranges over $\mathcal{L}_0(Atm)$. For notational convenience we write \mathcal{L}_0 instead of $\mathcal{L}_0(Atm)$, \mathcal{L}_0^T instead of $\mathcal{L}_0^T(Atm)$ and \mathcal{L} instead of $\mathcal{L}(Atm)$, when the context is unambiguous. The formula $\Box_{\mathfrak{m}} \alpha$ is read “agent \mathfrak{m} currently has the implicit belief that α ”. The other Boolean constructions \top , \perp , \vee , \rightarrow and \leftrightarrow are defined in the standard way. Notice that only formulas from the sublanguage \mathcal{L}_0 can be in the scope of the implicit belief operator $\Box_{\mathfrak{m}}$. Therefore, nesting of this operator is not allowed (e.g., $\Box_{\mathfrak{m}} \neg \Box_{\mathfrak{m}} p^t$ is not a well-formed formula). As we will show at the end of the section, this syntactic restriction on our language is useful to make the complexity of its satisfiability problem the same as the complexity of SAT.

The interpretation of the language \mathcal{L} exploits the notion of belief base. While the notions of possible world and epistemic alternative are primitive in the standard Kripke semantics for epistemic logic [Fagin *et al.* 1995], they are defined from the primitive concept of belief base in our semantics.

Definition 22 (State) *A state is a tuple $S = (B, V)$ where (i) $B \subseteq \mathcal{L}_0$ is agent \mathfrak{m} 's belief base (or, agent \mathfrak{m} 's subjective view of the actual situation), (ii) $V \subseteq \mathcal{L}_0^T$ is the actual situation, and such that, for every $t, t' \in \mathbb{N}$,*

$$now^{\geq 0} \in V, \tag{5.1}$$

$$\text{if } now^{\geq t} \in V \text{ and } t' \leq t \text{ then } now^{\geq t'} \in V, \tag{5.2}$$

$$now^{\geq t} \in V \text{ iff } now^{\geq t} \in B. \tag{5.3}$$

The set of all states is denoted by \mathbf{S} .

Conditions (5.1) and (5.2) in the previous definition guarantees time consistency, namely, that the current time should be at least 0 and that if the current time is at least t and $t' \leq t$, then it should be at least t' . Condition (5.3) captures agent \mathbf{m} 's time-knowledge, namely, the assumption that \mathbf{m} has complete information about the current time.

The sublanguage $\mathcal{L}_0(Atm)$ is interpreted w.r.t. states as follows.

Definition 23 (Satisfaction) Let $S = (B, V) \in \mathbf{S}$. Then:

$$\begin{aligned} S \models x &\iff x \in V, \\ S \models \neg\alpha &\iff S \not\models \alpha, \\ S \models \alpha_1 \wedge \alpha_2 &\iff S \models \alpha_1 \text{ and } S \models \alpha_2, \\ S \models \Delta_{\mathbf{m}}\alpha &\iff \alpha \in B. \end{aligned}$$

Observe in particular the set-theoretic interpretation of the explicit belief operator for agent \mathbf{m} : agent \mathbf{m} actually has the explicit belief that α if and only if α is included in her actual belief base. This highlights the asymmetry between agent \mathbf{m} and agent \mathbf{h} in our semantics. We adopt agent \mathbf{m} 's *internal* perspective, that is, the point of view of its belief base.⁴ On the contrary, agent \mathbf{h} 's explicit beliefs are modeled from an *external* point of view and semantically interpreted in the same way as the other timed formulas in $\mathcal{L}_0^T(Atm)$.

A multi-agent belief model (MAB) is defined to be a state supplemented with a set of states, called *context*. The latter includes all states that are compatible with agent \mathbf{m} 's background knowledge.

Definition 24 (Model) A model is a pair (S, Cxt) , where $S \in \mathbf{S}$ and $Cxt \subseteq \mathbf{S}$. The class of all models is denoted by \mathbf{M} .

Note that we do not impose that $S \in Cxt$. When $Cxt = \mathbf{S}$ then (S, Cxt) is said to be *complete*, since \mathbf{S} is conceivable as the complete (or universal) context which contains all possible states.

Definition 25 (Epistemic alternatives) We define \mathcal{R} to be the binary relation on the set \mathbf{S} such that, for all $S = (B, V), S' = (B', V') \in \mathbf{S}$:

$$S\mathcal{R}S' \text{ if and only if } \forall \alpha \in B : S' \models \alpha.$$

$S\mathcal{R}S'$ means that S' is an epistemic alternative for the artificial agent \mathbf{m} at S . So \mathbf{m} 's set of epistemic alternatives at S , noted $\mathcal{R}(S) = \{S' \in \mathbf{S} : S\mathcal{R}S'\}$, includes exactly those states that satisfy \mathbf{m} 's explicit beliefs.

Definition 26 extends Definition 23 to the full language \mathcal{L} . Its formulas are interpreted with respect to models. We omit Boolean cases that are defined in the usual way.

⁴See [Aucher 2012] for an in-depth logical analysis of the internal perspective on modeling knowledge and belief.

Definition 26 (Satisfaction, cont.) *Let $(S, Cxt) \in \mathbf{M}$. Then:*

$$\begin{aligned} (S, Cxt) \models \alpha &\iff S \models \alpha; \\ (S, Cxt) \models \Box_m \varphi &\iff \forall S' \in Cxt, \text{ if } S\mathcal{R}S' \text{ then } (S', Cxt) \models \varphi. \end{aligned}$$

A formula $\varphi \in \mathcal{L}$ is valid in the class \mathbf{M} , noted $\models_{\mathbf{M}} \varphi$, if and only if $(S, Cxt) \models \varphi$ for every $(S, Cxt) \in \mathbf{M}$; it is satisfiable in \mathbf{M} if and only if $\neg\varphi$ is not valid in \mathbf{M} . As the following theorem indicates, the satisfiability problem for $\mathcal{L}(Atm)$ has the same complexity as SAT.

Theorem 10 *Checking satisfiability of $\mathcal{L}(Atm)$ formulas in the class \mathbf{M} is an NP-complete problem.*

SKETCH OF PROOF. Hardness is clear since $\mathcal{L}(Atm)$ extends the propositional logic language. As for membership, we can find a polysize satisfiability preserving translation from $\mathcal{L}(Atm)$ to propositional logic. The translation is divided in three steps. First, we transform the input formula in $\mathcal{L}(Atm)$ into negated normal form (NNF). Secondly, we translate the formula in NNF into a restricted mono-modal language with no nesting of the modal operator. Thirdly, we translate the latter into a propositional logic language in a way similar to the standard translation of modal logic into FOL. We take care of translating a finite theory including axioms corresponding to the four constraints of Definition 22. The axioms have the following form: $now^{\geq 0}$, $now^{\geq t} \rightarrow now^{\geq t'}$ for $t' \leq t$, $now^{\geq t} \leftrightarrow \Delta_m now^{\geq t}$ and $\neg now^{\geq t} \leftrightarrow \Delta_m \neg now^{\geq t}$. The theory is finite since we only need to consider instances of the axioms whose symbols occur in the input formula. For example, if $t' \leq t$ and both $now^{\geq t}$ and $now^{\geq t'}$ occur in the input formula, then $now^{\geq t} \rightarrow now^{\geq t'}$ should be included in the theory, otherwise not. See Appendix A for a detailed proof. ■

5.3.2 Dynamic extension

Let us now move from a static to a dynamic perspective by presenting an extension of the language $\mathcal{L}(Atm)$ with belief expansion operators. Specifically, we introduce the following language $\mathcal{L}^+(Atm)$, or simply \mathcal{L}^+ :

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Box_m \alpha \mid [+^t_m \alpha]\varphi,$$

where α ranges over \mathcal{L}_0 and t ranges over \mathbb{N} . The formula $[+^t_m \alpha]\varphi$ is read “ φ holds after agent m has privately learned that α and that the current time is at least t ” or simply “ φ holds after agent m has privately learned that α at time at least t ”.

Our extension has the following semantics relative to a model:

Definition 27 (Satisfaction relation, cont.) *Let $S = (B, V) \in \mathbf{S}$ and $(S, Cxt) \in \mathbf{M}$. Then,*

$$(S, Cxt) \models [+^t_m \alpha]\varphi \iff (S^{+^t_m \alpha}, Cxt) \models \varphi$$

with

$$\begin{aligned} S^{+t_m\alpha} &= (B^{+t_m\alpha}, V^{+t_m\alpha}), \\ V^{+t_m\alpha} &= V \cup \{now^{\geq t'} : t' \leq t\}, \\ B^{+t_m\alpha} &= B \cup \{\alpha\} \cup \{now^{\geq t'} : t' \leq t\}. \end{aligned}$$

Intuitively speaking, agent \mathbf{m} 's private learning that α at time at least t simply consists of (i) adding the information α to \mathbf{m} 's belief base, and (ii) moving the objective time and \mathbf{m} 's subjective view of time to index t .

As the following proposition indicates, the dynamic semantics given in Definition 27 is well-defined, as it guarantees that the structure resulting from a belief expansion operation belongs to the class \mathbf{M} , if the initial structure also belongs to \mathbf{M} .

Proposition 5 *Let $(S, Cxt) \in \mathbf{M}$. Then, $(S^{+t_m\alpha}, Cxt) \in \mathbf{M}$.*

Satisfiability and validity of formulas in \mathcal{L}^+ relative to the class \mathbf{M} are analogous to satisfiability and validity for formulas in $\mathcal{L}(Atm)$ defined above. Interestingly, adding belief expansion operators to the language \mathcal{L} does not increase the complexity of the corresponding satisfiability problem.

Theorem 11 *Checking satisfiability of $\mathcal{L}^+(Atm)$ formulas in the class \mathbf{M} is an NP-complete problem.*

SKETCH OF PROOF. The theorem is a consequence of Theorem 10 and the fact that we can find a polysize reduction of the satisfiability problem for $\mathcal{L}^+(Atm)$ to the satisfiability problem for $\mathcal{L}(Atm)$. The reduction makes use of reduction axioms which allow us to eliminate dynamic operators from the input formula and to obtain a logically equivalent formula in $\mathcal{L}(Atm)$. See Appendix A for a detailed proof. ■

It is useful to define the concept of logical consequence for the language $\mathcal{L}^+(Atm)$ which will be used at a later stage to define the action selection problem for the artificial agent \mathbf{m} . Let Σ be a finite subset of $\mathcal{L}_0(Atm)$ and let $\varphi \in \mathcal{L}^+(Atm)$. We say that φ is a logical consequence of Σ in the class \mathbf{M} , noted $\Sigma \models_{\mathbf{M}} \varphi$, if and only if, for every $(B, Cxt) \in \mathbf{M}$ such that $Cxt \subseteq \mathbf{S}(\Sigma)$ we have $(B, Cxt) \models \varphi$, with $\mathbf{S}(\Sigma) = \{B \in \mathbf{S} : \forall \alpha \in \Sigma, B \models \alpha\}$. We say that φ is Σ -satisfiable in the class \mathbf{M} if and only if, $\neg\varphi$ is not a logical consequence of Σ in \mathbf{M} . Clearly, φ is valid if and only if φ is a logical consequence of \emptyset , and φ is satisfiable if and only if φ is \emptyset -satisfiable.

As the following deduction theorem indicates, the logical consequence problem with a finite set of premises can be reduced to the satisfiability problem.

Theorem 12 *Let $\varphi \in \mathcal{L}^+(Atm)$ and let $\Sigma \subset \mathcal{L}_0(Atm)$ be finite. Then, $\Sigma \models_{\mathbf{M}} \varphi$ if and only if $\models_{\mathbf{M}} \bigwedge_{\alpha \in \Sigma} \Box_{\mathbf{m}} \alpha \rightarrow \varphi$.*

5.4 Artificial agent architecture

In this section, we are going to show how the formal language presented in Section 5.3 can be used to endow the artificial agent \mathbf{m} with the capacity (i) to select an executable action for achieving a certain goal, and (ii) to revise its beliefs during its interaction with agent \mathbf{h} .

5.4.1 Action selection

Let $Act_{\mathbf{m}} = \{+_{\mathbf{m}}^t\alpha : \alpha \in \mathcal{L}_0, t \in \mathbb{N}\}$ be the set of belief expansion events, or informative actions, formally defined in Section 5.3.2. Such informative actions have executability preconditions that are specified by the following function $\mathcal{P} : Act_{\mathbf{m}} \rightarrow \mathcal{L}(Atm)$. We define the following operator of successful occurrence of an event in $Act_{\mathbf{m}}$:

$$\langle\langle +_{\mathbf{m}}^t\alpha \rangle\rangle\varphi \stackrel{\text{def}}{=} \mathcal{P}(+_{\mathbf{m}}^t\alpha) \wedge [+_{\mathbf{m}}^t\alpha]\varphi.$$

The formula $\langle\langle +_{\mathbf{m}}^t\alpha \rangle\rangle\varphi$ in $\mathcal{L}(Atm)$ has to be read “agent \mathbf{m} can privately learn that α at time t and φ holds after the occurrence of this learning event”.

Suppose the artificial agent \mathbf{m} has a goal represented by a formula in the language $\mathcal{L}_0(Atm)$. An action selection problem for \mathbf{m} just consists of finding an executable action in its finite action repertoire whose execution guarantees that the goal will be achieved.

Definition 28 (Action selection problem) *An action selection problem is a tuple $\langle \Sigma, Op, \alpha_G \rangle$ where:*

- $\Sigma \subset \mathcal{L}_0(Atm)$ is a finite set of agent \mathbf{m} 's available information,
- $Op \subset Act_{\mathbf{m}}$ is a finite set of operators representing agent \mathbf{m} 's action repertoire,
- $\alpha_G \in \mathcal{L}_0(Atm)$ is agent \mathbf{m} 's goal.

A solution to the action selection problem $\langle \Sigma, Op, \alpha_G \rangle$ is an action ϵ in Op such that $\Sigma \models_{\mathbf{M}} \langle\langle \epsilon \rangle\rangle \Box_{\mathbf{m}}\alpha_G$. In other words, a solution to the action selection problem is an executable action in agent \mathbf{m} 's repertoire such that if agent \mathbf{m} has the information in Σ at its disposal then, after executing the action, it will believe that its goal α_G is achieved.

An action selection problem can be seen as a limit case of a planning problem with a single action to be selected from the action repertoire, instead of a sequence of actions as in the general planning domain. Thanks to Theorems 11 and 12, we can conclude that the action selection problem is NP-complete too.

5.4.2 Belief change

A crucial component of the action selection problem defined in Definition 28 is agent \mathbf{m} 's available information Σ . The latter includes the information about the rules of

the game as well as information about the actual configuration of the game and agent \mathfrak{h} 's beliefs. Some of this information evolve during the game. In this section we describe agent \mathfrak{m} 's belief change mechanisms that are responsible for inducing this kind of information change.

We distinguish belief update from belief revision. Belief update consists of making the agent \mathfrak{m} 's explicit beliefs evolve from a time t to the next time $t + 1$. Belief revision is the result of agent \mathfrak{m} learning a new fact and adding a new piece of information to its belief base. We assume agent \mathfrak{m} 's available information Σ used in the action selection phase is split into two sets $\Sigma_c, \Sigma_m \subseteq \mathcal{L}_0$. They denote, respectively, the core (or, immutable) information in agent \mathfrak{m} 's belief base and the volatile (or, mutable) information in agent \mathfrak{m} 's belief base. Agent \mathfrak{m} 's core beliefs are stable and do not change under belief revision. On the contrary, volatile beliefs can change due to a belief revision operation. Moreover, we assume the mutable belief base Σ_m is the union of two sets Σ_h and Σ_f . Σ_h includes all hypothetical information that agent \mathfrak{m} can use for reasoning, while Σ_f contains agent \mathfrak{m} 's factual information, that is, all facts that agent \mathfrak{m} observes during its interaction with agent \mathfrak{h} through the game. We assume information in Σ_f has priority over information in Σ_h , in the sense that in case its belief base becomes inconsistent, agent \mathfrak{m} prefers to remove information from Σ_h than to remove information from Σ_f in order to restore consistency.

We assume agent \mathfrak{m} 's mutable belief base is *present-focused*, in the sense that it contains information about the present time point t and no other time point t' in the future or in the past of t . Let us define this notion formally. The following function specifies the set of time indexes appearing in a \mathcal{L}_0 -formula:

$$\begin{aligned} \text{time}(p^t) &= \{t\}, \\ \text{time}(\Delta_{\mathfrak{h}}^t \alpha) &= \{t\} \cup \text{time}(\alpha), \\ \text{time}(\text{now}^{\geq t}) &= \{t\}, \\ \text{time}(\neg \alpha) &= \text{time}(\alpha), \\ \text{time}(\alpha_1 \wedge \alpha_2) &= \text{time}(\alpha_1) \cup \text{time}(\alpha_2). \end{aligned}$$

The fact that Σ_m is *present-focused* means that there exists $\Sigma \subseteq \mathcal{L}_0$ and $t \in \mathbb{N}$ such that:

- $\forall \alpha \in \Sigma, \text{time}(\alpha) = \{t\}$,
- $\Sigma_m = \Sigma \cup \{\text{now}^{\geq t'} : t' \leq t\}$.

5.4.2.1 Belief update

We see belief update as a function Upd which takes a triple $(\Sigma_c, \Sigma_h, \Sigma_f)$ specifying the core belief base, the hypothetical mutable belief base and the factual mutable belief base of agent \mathfrak{m} as input and returns a new triple $(\Sigma'_c, \Sigma'_h, \Sigma'_f)$ as output.

We consider a specific update function which simply increments of one unit the time indexes of all formulas appearing in agent \mathfrak{m} 's mutable belief base, while keeping

agent \mathbf{m} 's core belief base unchanged. That is, let $incrt$ be the function devoted to increment the time indexes of a \mathcal{L}_0 -formula of one unit:

$$\begin{aligned} incrt(p^t) &= p^{t+1}, \\ incrt(\Delta_{\mathfrak{h}}^t \alpha) &= \Delta_{\mathfrak{h}}^{t+1} incrt(\alpha), \\ incrt(\Delta_{\mathfrak{m}} \alpha) &= \Delta_{\mathfrak{m}} incrt(\alpha), \\ incrt(now^{\geq t}) &= now^{\geq t+1}, \\ incrt(\neg \alpha) &= \neg incrt(\alpha), \\ incrt(\alpha_1 \wedge \alpha_2) &= incrt(\alpha_1) \wedge incrt(\alpha_2). \end{aligned}$$

For each finite $X \subseteq \mathcal{L}_0$, we define $incrt(X) = \{incrt(\alpha) : \alpha \in X\}$.

We stipulate that $Upd(\Sigma_c, \Sigma_h, \Sigma_f) = (\Sigma'_c, \Sigma'_h, \Sigma'_f)$ if and only if:

- $\Sigma'_c = \Sigma_c$,
- $\Sigma'_h = incrt(\Sigma_h)$,
- $\Sigma'_f = incrt(\Sigma_f) \cup \{now^{\geq 0}\}$.

It is straightforward to verify that $\Sigma'_m = \Sigma'_h \cup \Sigma'_f$ is present-focused since $\Sigma_m = \Sigma_h \cup \Sigma_f$ is present-focused too.

The belief update function so defined relies on two assumptions that make perfect sense in the context of the interaction between agent \mathbf{m} and agent \mathfrak{h} in the Yōkai game. First, agent \mathbf{m} does not keep in its memory information about past facts. We make this assumption since we want to maintain agent \mathbf{m} 's mutable belief base manageable and to avoid that it constantly increases through the game. Secondly, agent \mathbf{m} assumes that *by default* (i) the world does not change and agents \mathfrak{h} and \mathbf{m} do not forget what they believe, and (ii) agents \mathfrak{h} and \mathbf{m} have common knowledge that (i).⁵ Nonetheless, after having updated its belief base, agent \mathbf{m} can learn new information which is incompatible with its actual beliefs. In this case, it will need to revise its beliefs. How agent \mathbf{m} must revise its beliefs is the content of the next section.

5.4.2.2 Belief revision

At each time step of the game agent \mathbf{m} performs belief revision after belief update. We assume that in the belief revision phase formulas in the language \mathcal{L}_0 are treated as atomic formulas. In particular, let \mathcal{L}_{PROP} be the propositional language built from the following set of atomic propositions:

$$PROP = \{\tau_x : x \in \mathcal{L}_0^T(Atm)\} \cup \{\tau_{\Delta_{\mathfrak{m}} \alpha} : \Delta_{\mathfrak{m}} \alpha \in \mathcal{L}_0(Atm)\}.$$

⁵'By default' means "if no agent acts".

The following translation transforms each formula in \mathcal{L}_0 into its propositional logic counterpart in the language $\mathcal{L}_{\text{PROP}}$:

$$\begin{aligned} tr_{\text{PROP}}(x) &= \tau_x \text{ for } x \in \mathcal{L}_0^T(\text{Atm}), \\ tr_{\text{PROP}}(\Delta_m \alpha) &= \tau_{\Delta_m \alpha}, \\ tr_{\text{PROP}}(\neg \alpha) &= \neg tr_{\text{PROP}}(\alpha), \\ tr_{\text{PROP}}(\alpha_1 \wedge \alpha_2) &= tr_{\text{PROP}}(\alpha_1) \wedge tr_{\text{PROP}}(\alpha_2). \end{aligned}$$

For each finite $X \subseteq \mathcal{L}_0$, we define $tr_{\text{PROP}}(X) = \{tr_{\text{PROP}}(\alpha) : \alpha \in X\}$.

We say that X is propositionally consistent if and only if $\perp \notin Cn(tr_{\text{PROP}}(X))$, where Cn is the classical deductive closure operator over the propositional language $\mathcal{L}_{\text{PROP}}$. Clearly, the latter is equivalent to saying that $\bigwedge_{\alpha \in X} tr_{\text{PROP}}(\alpha)$ is satisfiable in propositional logic.

Let $\Sigma_{\text{input}} \subseteq \mathcal{L}_0$ be agent m 's input information set. We see belief revision as a function Rev which takes a quadruple $(\Sigma_c, \Sigma_h, \Sigma_f, \Sigma_{\text{input}})$ specifying the core belief base, the hypothetical mutable belief base and the factual mutable belief base of agent m together with the input information set and returns a triple $(\Sigma'_c, \Sigma'_h, \Sigma'_f)$.

The revision of $(\Sigma_c, \Sigma_h, \Sigma_f)$ by input Σ_{input} , noted $Rev(\Sigma_c, \Sigma_h, \Sigma_f, \Sigma_{\text{input}})$, is formally defined as follows:

1. if $\Sigma_c \cup \Sigma_{\text{input}}$ is not propositionally consistent then $Rev(\Sigma_c, \Sigma_h, \Sigma_f, \Sigma_{\text{input}}) = (\Sigma_c, \Sigma_h, \Sigma_f)$,
2. otherwise, $Rev(\Sigma_c, \Sigma_h, \Sigma_f, \Sigma_{\text{input}}) = (\Sigma'_c, \Sigma'_h, \Sigma'_f)$, with

$$\begin{aligned} \Sigma'_c &= \Sigma_c, \\ \Sigma'_f &= \bigcap_{X \in MCS(\Sigma_c, \Sigma_f, \Sigma_{\text{input}})} X, \\ \Sigma'_h &= \bigcap_{X \in MCS(\Sigma_c \cup \Sigma'_f, \Sigma_h, \emptyset)} X, \end{aligned}$$

where for all $\Sigma, \Sigma', \Sigma'' \subseteq \mathcal{L}_0$, we have $X \in MCS(\Sigma, \Sigma', \Sigma'')$ if and only if:

- $X \subseteq \Sigma' \cup \Sigma''$,
- $\Sigma'' \subseteq X$,
- $X \cup \Sigma$ is propositionally consistent, and
- there is no $X' \subseteq \Sigma' \cup \Sigma''$ such that $X \subset X'$ and $X' \cup \Sigma$ is propositionally consistent.

The revision function Rev has the following effects on agent m 's beliefs. First of all, the core belief base is not modified.

Secondly, the input Σ_{input} is added to the factual mutable belief base only if it is consistent with the core beliefs. In the latter case, the updated factual mutable

belief base is equal to the intersection of the subsets of the factual mutable belief base which are maximally consistent with respect to the core belief base and which include the input Σ_{input} . This guarantees that belief revision satisfies minimal change for factual information.

Finally, agent \mathbf{m} checks whether the hypotheses in its hypothetical mutable belief base are still consistent with its core beliefs and its revised factual information. If so, it does not modify them. If not, it minimally contracts its hypothetical mutable belief base: it takes the intersection of the subsets of the hypothetical mutable belief base which are maximally consistent with respect to the core belief base and its revised factual information.

For notational convenience, we write $Rev^{core}(\Sigma_c, \Sigma_h, \Sigma_f)$ to denote Σ'_c , $Rev^{hyp}(\Sigma_c, \Sigma_h, \Sigma_f)$ to denote Σ'_h and $Rev^{fact}(\Sigma_c, \Sigma_h, \Sigma_f)$ to denote Σ'_f . Note that if $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_f$ is propositionally consistent, then $Rev^{core}(\Sigma_c, \Sigma_h, \Sigma_f) \cup Rev^{hyp}(\Sigma_c, \Sigma_h, \Sigma_f) \cup Rev^{fact}(\Sigma_c, \Sigma_h, \Sigma_f)$ is propositionally consistent too.

In the next section, we will provide a formalization of the Yōkai board-game with the aid of the language $\mathcal{L}^+(Atm)$. We will represent agent \mathbf{m} 's action repertoire to be used in the action selection problem as a set of events in $Act_{\mathbf{m}}$ affecting \mathbf{m} 's beliefs. We assume in its turn agent \mathbf{m} faces four consecutive action selection problems. First, agent \mathbf{m} has to decide at which card to look. It faces this problem twice. Then, it has to decide which card to move and to which position. Finally, it has to decide whether to activate a hint or mark a card with a hint. For every action of \mathbf{m} , we will specify the corresponding executability precondition.

Moreover, we will specify agent \mathbf{m} 's available information at the beginning of the game and clearly distinguish mutable (factual and hypothetical) information from core information.

5.5 Game modeling

In this section, we first formalize all the static aspects of the game (the different rules, the initial state of the game, etc.) and then the action representation. Finally, we present an example of action selection by agent \mathbf{m} in the game.

5.5.1 Static aspects

Let be the following sets:

$$\begin{aligned}
TIME &= \{0, 1, \dots, end\} \text{ where } end = 56 \\
TIME^* &= TIME \setminus \{0\} \\
GRID &= \{1, \dots, 32\} \times \{1, \dots, 32\}, \\
IPOS &= \{(l, c) \in GRID : l, c \in \{15, \dots, 18\}\}, \\
COLORS &= \{r, g, b, y\}, \\
HINTS &= 2^{COLORS} \setminus \{\{\}, \{r, g, b, y\}\}, \\
CARDS &= \{1, \dots, 16\} \\
CARDS^n &= \{X \in 2^{CARDS} : |X| = n\} \text{ with } n \in \mathbb{N}
\end{aligned}$$

There are seven hints at the start of the game, and each player must take turns activating a hint or using a hint to mark a card. Thus, each player plays 7 times. As each player must take 4 different actions in turn, the game lasts a maximum of $(7 \times 4) \times 2 = 56$ time units (56 actions are executed during a game). So, let $TIME$ be the set of time points, including initial state of the game at time 0. End of the game is marked by the natural $end \in TIME$ and $end = 56$.

However, each player only performs the action of moving a card once among the 4 actions she/it must perform during her/its turn. There are therefore a total of $7 \times 2 = 14$ card moves. As in the initial state the 16 cards are placed in a square of 4×4 cards, whatever the movements made during a game, the cards will evolve in a grid of 32×32 positions represented by the set $GRID$, and $IPOS$ represents the set of cards positions at the start of the game.

A hint is viewed as a non empty subset of 1, 2 or 3 colors among 4 different colors (r for red, g for green, b for blue and y for yellow). $HINTS$ is the set of all hints and $|HINTS| = 14$. When the game starts, only 7 hints are available.

Vocabulary. The set of atomic proposition Atm is defined as follows:

$$\begin{aligned}
Atm = & \bigcup_{\substack{t \in TIME \\ x \in CARDS \\ c \in COLORS}} \{col_{x,c}^t\} \cup \bigcup_{\substack{t \in TIME \\ x \in CARDS \\ p \in GRID}} \{pos_{x,p}^t\} \cup \bigcup_{\substack{t \in TIME \\ h \in HINTS}} \{active_h^t\} \cup \\
& \bigcup_{\substack{t \in TIME \\ x \in CARDS \\ h \in HINTS}} \{mark_{x,h}^t\} \cup \bigcup_{\substack{t \in TIME \\ x \in CARDS \\ p \in GRID}} \{legMov_{x,p}^t\}
\end{aligned}$$

where: $col_{x,c}^t$ is true iff card x is of color c at time t ; $pos_{x,p}^t$ is true iff card x is at position p at time t ; $active_h^t$ is true iff hint h is enabled at time t ; $mark_{x,h}^t$ is true iff card x is marked with hint h at time t . Finally, $legMov_{x,p}^t$ is true iff to move card x to position p at time t is legal (see paragraph “The separation constraint” page 91 for more details).

Moreover, we define a function $\sigma : CARDS \rightarrow IPOS$ that assigns to each card x a position p in the initial positions set, and a function $NEIG : GRID \rightarrow 2^{GRID}$ that assigns to each position (l, c) the set of its neighboring positions in the grid $NEIG((l, c)) = \{(l+1, c), (l-1, c), (l, c+1), (l, c-1)\} \cap GRID$.

In the following, we consider that belief bases of agents are defined in \mathcal{L}_0 and are divided into two (sub)bases: the *core beliefs* base (containing all the beliefs that cannot be modified during the game), and the *mutable beliefs* base (containing all the belief that may change). As we want to model the game from \mathbf{m} 's perspective, we only model its beliefs about the facts of the game or about the beliefs of agent \mathbf{h} .

Initial mutable beliefs of agent \mathbf{m} . Recall (see Section 5.4.2) that mutable beliefs base $\Sigma_m = \Sigma_f \cup \Sigma_h$ (union of factual beliefs base and hypothetical beliefs base).

In initial state, Σ_f (the factual beliefs base of agent \mathbf{m}) contains not only beliefs on facts of the world (which is captured by the definition of the set Σ_f^W in the next paragraph) but also on the beliefs of the agent \mathbf{h} on these facts of the world (see below the definition of Σ_f^h).

Σ_f^W contains all \mathbf{m} 's beliefs about the initial state of the game. It includes the fact that: time is at least 0, each card position is known (we assume that function σ is known), the cards color is not known, only seven hints are available et and the others are not known, none is activated yet, and none marks a card yet. Moreover, agent \mathbf{m} knows all positions that are adjacent and those that are not.

Σ_f^h includes the fact that agent \mathbf{m} believes that the initial state of the game is also known by agent \mathbf{h} . That is, \mathbf{m} believes that \mathbf{h} believes all the facts included in Σ_f^W , plus the fact that \mathbf{h} does not know the color of any card.

Finally, Σ_f is the union of Σ_f^W and Σ_f^h . So, formally:

$$\begin{aligned} \Sigma_f^W &= \{now^{\geq 0}\} \cup \bigcup_{x \in CARDS} \{pos_{x, \sigma(x)}^0\} \cup \bigcup_{\substack{x \in CARDS \\ c \in COLORS}} \{\neg \Delta_m col_{x,c}^0\} \cup \\ &\quad \bigcup_{h \in HINTS} \{\neg active_h^0\} \cup \bigcup_{\substack{x \in CARDS \\ h \in HINTS}} \{\neg mark_{x,h}^0\} \\ \Sigma_f^h &= \bigcup_{\alpha \in \Sigma_f^W} \{\Delta_h^0 \alpha\} \cup \bigcup_{\substack{x \in CARDS \\ c \in COLORS}} \{\neg \Delta_h^0 col_{x,c}^0\} \\ \Sigma_f &= \Sigma_f^W \cup \Sigma_f^h \end{aligned}$$

Note that Σ_f does not contain any conjunction or disjunction big operator. It is for technical considerations: when expanding the belief base of the agent following the execution of an action, formulas can be deleted from the base. If these are present in the form of a conjunction, it suffices that one term of this conjunction is inconsistent with the new information for the whole conjunction to be deleted. In order to minimize the suppressed information, each conjunction is seen as a set of

(sub-)formulas.

In initial state, the hypothetical beliefs base Σ_h of agent \mathbf{m} includes the fact that \mathbf{m} believes that when a card is marked with a hint that provides a set of possible colors for that card, this card cannot have a color other than those provided by the hint. That is:

$$\Sigma_h = \bigcup_{\substack{x \in \text{CARDS} \\ h \in \text{HINTS} \\ c \in \text{COLORS} \setminus h}} \{ \text{mark}_{x,h}^0 \rightarrow \neg \text{col}_{x,c}^0, \Delta_h^0 \text{mark}_{x,h}^0 \rightarrow \Delta_h^0 \neg \text{col}_{x,c}^0 \}$$

For example, $\Sigma_h \supseteq \{ \text{mark}_{1,\{g,r\}}^0 \rightarrow \neg \text{col}_{1,b}^0, \text{mark}_{1,\{g,r\}}^0 \rightarrow \neg \text{col}_{1,y}^0 \}$. Suppose now that $\text{mark}_{1,\{g,r\}}^0 \in \Sigma_f$, then $\Sigma_c \cup \Sigma_m \models \Box_{\mathbf{m}} ((\text{col}_{1,g}^0 \vee \text{col}_{1,r}^0) \wedge \neg \text{col}_{1,b}^0 \wedge \neg \text{col}_{1,y}^0)$ holds (see below for Σ_c definition).

Why is this type of knowledge in Σ_h and not in Σ_f ? Because it is general knowledge that can be questioned (like default rules of reasoning). Thus, if we suppose that agent \mathbf{h} puts a hint $h = \{g\}$ on a card (which means “this card is green”) then agent \mathbf{m} can deduce implicitly (thanks to Σ_h) that this card is green. But in fact, four cases are possible: 1) \mathbf{m} already explicitly believes that the card is green, in which case \mathbf{m} does not learn anything that it did not already know; 2) \mathbf{m} does not know the color of the card, and in this case it can implicitly deduce that the card is neither yellow, nor red, nor blue; 3) \mathbf{m} already explicitly believes that the card is of another color (red, for example), and in this case it is primarily Σ_h that will be revised; 4) \mathbf{m} implicitly believes that the card cannot be green (because \mathbf{m} already knows the 4 green cards for example), and then this case is treated as 3).

Initial core beliefs of agent \mathbf{m} . Core belief base includes all the agent \mathbf{m} 's beliefs that cannot change during the game. It concerns both integrity constraints (that describe rules of the game) and successor state axioms (SSAs) that describe the building of the belief base after the execution of an action. SSAs describe both what changes in the new state, and what do not (thanks to frame axioms). So, we

define the following integrity constraints (IC):

$$\Sigma_c^{ic} = \bigcup_{t \in TIME} \left\{ \bigwedge_{x \in CARDS} \bigvee_{p \in GRID} pos_{x,p}^t, \right. \quad (ICP1)$$

$$\bigwedge_{\substack{x \in CARDS \\ p, p' \in GRID: p \neq p'}} \neg(pos_{x,p}^t \wedge pos_{x,p'}^t), \quad (ICP2)$$

$$\bigwedge_{\substack{p \in GRID \\ x, x' \in CARDS: x \neq x'}} \neg(pos_{x,p}^t \wedge pos_{x',p}^t), \quad (ICP3)$$

$$\bigwedge_{x \in CARDS} \bigvee_{c \in COLORS} col_{x,c}^t, \quad (ICC4)$$

$$\bigwedge_{\substack{x \in CARDS \\ c, c' \in COLORS: c \neq c'}} \neg(col_{x,c}^t \wedge col_{x,c'}^t), \quad (ICC5)$$

$$\bigwedge_{c \in COLORS} \bigvee_{X \in CARDS^4} \left(\bigwedge_{x \in X} col_{x,c}^t \wedge \bigwedge_{x \notin X} \neg col_{x,c}^t \right), \quad (ICC6)$$

$$\bigwedge_{\substack{h \in HINTS \\ x, x' \in CARDS \\ x \neq x'}} \neg(mark_{x,h}^t \wedge mark_{x',h}^t), \quad (ICH7)$$

$$\bigwedge_{\substack{x \in CARDS \\ h \in HINTS}} \neg(active_h^t \wedge mark_{x,h}^t) \left. \right\} \quad (ICH8)$$

ICP1 and ICP2 mean that each card has at least one, and at most one, position respectively ; ICP3 means that each position can only accommodate one card. In the same way, ICC4 and ICC5 means that each card has at least one, and at most one, color respectively. ICC6 means that there are exactly 4 cards of each color. ICH7 means a hint cannot marks two different cards. ICH8 means that a hint is either not active and not marking, active and not marking, or not active and marking. Finally, Σ_c assumes that agent \mathfrak{m} believes that the initial state of the game is also known by agent \mathfrak{h} .

Note that by definition, Σ_c cannot be revised by any action, one can add conjunctions directly to this set (rather than the set of their sub-formulas).

Frame axioms (FA) describe the facts that does not change after the execution of an action. For convenience, we define the following FA abbreviations (for every

$X \subseteq CARDS$, $H \subseteq HINTS$ and $t \in TIME^*$):

$$\begin{aligned} \text{posFA}_X^t &\stackrel{\text{def}}{=} \bigwedge_{\substack{x \in CARDS \setminus X \\ p \in GRID}} (\text{pos}_{x,p}^t \leftrightarrow \text{pos}_{x,p}^{t-1}) \\ \text{colFA}_X^t &\stackrel{\text{def}}{=} \bigwedge_{\substack{x \in CARDS \setminus X \\ c \in COLORS}} \left((\text{col}_{x,c}^t \leftrightarrow \text{col}_{x,c}^{t-1}) \wedge (\neg \Delta_m \text{col}_{x,c}^t \leftrightarrow \neg \Delta_m \text{col}_{x,c}^{t-1}) \right) \\ \text{hintFA}_H^t &\stackrel{\text{def}}{=} \bigwedge_{h \in HINTS \setminus H} (\text{active}_h^t \leftrightarrow \text{active}_h^{t-1}) \wedge \\ &\quad \bigwedge_{\substack{x \in CARDS \\ h \in HINTS \setminus H}} (\text{mark}_{x,h}^t \leftrightarrow \text{mark}_{x,h}^{t-1}) \end{aligned}$$

posFA_X^t (resp. colFA_X^t) reads “the position (resp. color) of every cards except those in X is preserved from time $t - 1$ to t ”. $\text{hintFA}_X^t H$ reads “the status (active or marking) of every hints except those in H is preserved from time $t - 1$ to t ”.

Finally, we define the following successor state axioms that are useful for planning since they allow to deduce what will be true at some time t in the future:

$$\begin{aligned} \Sigma_c^{ssa} = \bigcup_{t \in TIME^*} \left\{ \right. & \\ & \bigwedge_{\substack{x \in CARDS \\ p \in GRID}} (\text{pos}_{x,p}^t \wedge \neg \text{pos}_{x,p}^{t-1} \rightarrow \text{posFA}_{\{x\}}^t \wedge \text{colFA}_{\emptyset}^t \wedge \text{hintFA}_{\emptyset}^t), \quad (\text{SSA9}) \\ & \bigwedge_{\substack{x \in CARDS \\ c \in COLORS}} (\text{col}_{x,c}^t \wedge \neg \Delta_m \text{col}_{x,c}^{t-1} \rightarrow \text{posFA}_{\emptyset}^t \wedge \text{colFA}_{\{x\}}^t \wedge \text{hintFA}_{\emptyset}^t), \quad (\text{SSA10}) \\ & \bigwedge_{h \in HINTS} (\text{active}_h^t \wedge \neg \text{active}_h^{t-1} \rightarrow \text{posFA}_{\emptyset}^t \wedge \text{colFA}_{\emptyset}^t \wedge \text{hintFA}_{\{h\}}^t), \quad (\text{SSA11}) \\ & \bigwedge_{\substack{x \in CARDS \\ h \in HINTS}} (\text{mark}_{x,h}^t \wedge \neg \text{mark}_{x,h}^{t-1} \rightarrow \text{posFA}_{\emptyset}^t \wedge \text{colFA}_{\emptyset}^t \wedge \text{hintFA}_{\{h\}}^t) \quad (\text{SSA12}) \\ & \left. \right\} \end{aligned}$$

SSA9 means that if card x has a new position p at time t , the position of others cards, the color of every cards, and the status of all hints, remain unchanged from $t - 1$ to t . **SSA10** means that if card x has a new color c at time t , the position of every cards, the color of every other cards, and the status of all hints, remain unchanged from $t - 1$ to t . **SSA11** and **SSA12** mean that if a hint becomes active (resp. marks a card) between time $t - 1$ and t then neither positions and colors of cards did change nor the status of other hints. Moreover, we assume that at least one action is performed at each time point. Formally, this is represented by the disjunction of antecedents of **SSA9** to **SSA12**.

Finally, we suppose that agent \mathbf{m} believes that all the fact in its core belief base

are also known by the other agent \mathfrak{h} .

$$\Sigma_c = \Sigma_c^{ic} \cup \Sigma_c^{ssa}$$

In principle, agent \mathfrak{m} believes that agent \mathfrak{h} shares the same rules of the game (set Σ_c^{ic}) and the same successor states axioms (set Σ_c^{ssa}), but for the sake of simplicity, we omit this part which is not currently used by \mathfrak{m} for planning actions.

5.5.2 Dynamic aspects

We introduce in what follows actions of perceptions. These actions allow the agent \mathfrak{m} to expand its belief base by a certain formula.

Vocabulary. For convenience, we define the action repertoire $ACT \subseteq EVT$ of agent \mathfrak{m} (for every $t < end$, $x \in CARDS$, $p \in GRID$, $h \in HINTS$):

$$\begin{aligned} +_{\mathfrak{m}}^{col_{x,c}^{t+1}} &\stackrel{\text{def}}{=} +_{\mathfrak{m}}^{t+1} \left(col_{x,c}^{t+1} \wedge \Delta_{\mathfrak{m}} col_{x,c}^{t+1} \wedge \right. \\ &\quad \left. (\Delta_{\mathfrak{h}}^{t+1} \bigvee_{c' \in COLORS} \Delta_{\mathfrak{m}} col_{x,c'}^{t+1}) \wedge \Delta_{\mathfrak{h}}^{t+1} now^{\geq t+1} \right) \\ +_{\mathfrak{m}}^{pos_{x,p}^{t+1}} &\stackrel{\text{def}}{=} +_{\mathfrak{m}}^{t+1} \left(pos_{x,p}^{t+1} \wedge \Delta_{\mathfrak{m}} pos_{x,p}^{t+1} \wedge \Delta_{\mathfrak{h}}^{t+1} pos_{x,p}^{t+1} \wedge \Delta_{\mathfrak{h}}^{t+1} now^{\geq t+1} \right) \\ +_{\mathfrak{m}}^{actHint_h^{t+1}} &\stackrel{\text{def}}{=} +_{\mathfrak{m}}^{t+1} \left(active_h^{t+1} \wedge \Delta_{\mathfrak{m}} active_h^{t+1} \wedge \Delta_{\mathfrak{h}}^{t+1} active_h^{t+1} \wedge \right. \\ &\quad \left. \Delta_{\mathfrak{h}}^{t+1} now^{\geq t+1} \right) \\ +_{\mathfrak{m}}^{markHint_{x,h}^{t+1}} &\stackrel{\text{def}}{=} +_{\mathfrak{m}}^{t+1} \left(mark_{x,h}^{t+1} \wedge \Delta_{\mathfrak{m}} mark_{x,h}^{t+1} \wedge \Delta_{\mathfrak{h}}^{t+1} mark_{x,h}^{t+1} \wedge \right. \\ &\quad \left. \Delta_{\mathfrak{h}}^{t+1} now^{\geq t+1} \right) \end{aligned}$$

As each action entails a new time point, each action execution entails the fact that the agent learns not only that time increases (which is taken into account directly in the semantics, see Definition 22), but also that time increases for agent \mathfrak{h} (so, $\Delta_{\mathfrak{h}}^{t+1} now^{\geq t+1}$ must be added to \mathfrak{m} 's belief base). Moreover, each time p^t is added, $\Delta_{\mathfrak{m}} p^t$ is also added. The reason is that in the case where the agent does not explicitly believe that p ($\neg \Delta_{\mathfrak{m}} p^t$) and we want to add p^t to the belief base of the agent, we have to remove $\neg \Delta_{\mathfrak{m}} p^t$ from the base. For example, in the initial state, the agent does not know the color of the card x (that is, for any color c , $\neg \Delta_{\mathfrak{m}} col_{x,c}^0$ belongs to the base). As soon as he observes the color of a card (which is green, for example), then $\neg \Delta_{\mathfrak{m}} col_{x,g}^1$ must be deleted from the base, which will be done automatically by our revision module as soon as $\Delta_{\mathfrak{m}} col_{x,g}^1$ is added to the database (in addition to $col_{x,g}^1$).

Note also that $\Delta_{\mathfrak{h}}^{t+1} \bigvee_{c' \in COLORS} \Delta_{\mathfrak{m}} col_{x,c'}^{t+1}$ reads “agent \mathfrak{h} believes that \mathfrak{m} knows the color of card c' ”, that is quite different from $\Delta_{\mathfrak{h}}^{t+1} \Delta_{\mathfrak{m}} \bigvee_{c' \in COLORS} col_{x,c'}^{t+1}$ that reads “agent \mathfrak{h} believes that agent \mathfrak{m} believes that card x has at least one color (among $COLORS$)”.

Moreover, each action about a fact also informs agent \mathfrak{m} that \mathfrak{h} believes this fact.

Finally, note that these actions concern the point of view of agent \mathbf{m} but say nothing about their author (which can be \mathbf{m} or \mathbf{h}).

The separation constraint. When moving a card from a position p to a position p' , it is forbidden to create two separate groups of cards (see Figure 5.1b). In other words, there must be a path between a given card x and all the other cards y , which automatically ensures that there is a path between any two different cards y' and y'' of the game.

We have a logical characterization of this constraint (See in Appendix A the detailed function). However, because this function is complex, for the sake of simplicity and for efficiency, we introduced in *Atm* definition (see paragraph “Vocabulary” page 85) an atomic formula $legMov_{x,p}^t$ whose truth value is supposed to be updated at a metalogical level.

The meaning of “ $legMov_{x,p}^t$ is true” is: “the move of card x towards position p at time t is authorized by the rules of the game”, that is: p is currently an empty position and there is a sequence of adjacent positions between p and any other occupied positions (except the initial position p'' of card x) through the set of currently occupied positions (excluding p'').

Action preconditions. We assume now that the operators in *ACT* have the following executability preconditions, for $t \in TIME$ such that $t < end$, $x \in CARDS$, $c \in COLORS$, $h \in HINTS$ and $now^=t \stackrel{\text{def}}{=} now^{\geq t} \wedge \neg now^{\geq t+1}$:

$$\begin{aligned} \mathcal{P}(+_m^{col_{x,c}^{t+1}}) &\stackrel{\text{def}}{=} now^=t \wedge \square_m \bigwedge_{h \in HINTS} \neg mark_{x,h}^t \\ \mathcal{P}(+_m^{pos_{x,p}^{t+1}}) &\stackrel{\text{def}}{=} now^=t \wedge \square_m \bigwedge_{h \in HINTS} \neg mark_{x,h}^t \wedge \square_m legMov_{x,p}^t \\ \mathcal{P}(+_m^{actHint_h^{t+1}}) &\stackrel{\text{def}}{=} now^=t \\ \mathcal{P}(+_m^{markHint_{x,h}^{t+1}}) &\stackrel{\text{def}}{=} now^=t \wedge \square_m active_h^t \wedge \bigvee_{c \in h} \square_m col_{x,c}^t \wedge \\ &\quad \square_m \bigwedge_{\substack{h' \in HINTS \\ h' \neq h}} \neg mark_{x,h'}^t \end{aligned}$$

$\mathcal{P}(+_m^{col_{x,c}^{t+1}})$ reads “agent \mathbf{m} implicitly believes that time is currently equal to t and also implicitly believes that no hint marks card x ”; $\mathcal{P}(+_m^{pos_{x,p}^{t+1}})$ reads “agent \mathbf{m} implicitly believes that time is currently equal to t and also implicitly believes that no hint marks card x and that it is currently authorized, w.r.t. rules of the game, to move cards x to position p ”; $\mathcal{P}(+_m^{actHint_h^{t+1}})$ reads “agent \mathbf{m} implicitly believes that time is currently equal to t ”; $\mathcal{P}(+_m^{markHint_{x,h}^{t+1}})$ reads “agent \mathbf{m} implicitly believes that time is currently equal to t , that hint h is currently active, that h includes the colors of card x , and that no other hint h' already marks card x ”.

5.5.3 Example of action selection

In this section, we are going to illustrate the action selection problem defined in Section 5.4.1 on a specific configuration of the game. We assume is agent \mathbf{m} 's turn to play and its goal α_G is to mark a card which is not actually marked, whose color is not known by the human and which is adjacent to a card of the same color. That is,

$$\alpha_G \stackrel{\text{def}}{=} \bigvee_{\substack{x \in \text{CARDS} \\ p \in \text{GRID} \\ c \in \text{COLORS}}} \left(\text{pos}_{x,p}^t \wedge \text{col}_{x,c}^t \wedge \neg \Delta_{\mathfrak{h}}^t \text{col}_{x,c}^t \wedge \bigwedge_{h \in \text{HINTS}} \neg \text{mark}_{x,h}^t \wedge \bigvee_{\substack{p' \in \text{NEIG}(p) \\ x' \in \text{CARDS}}} (\text{pos}_{x',p'}^t \wedge \text{col}_{x',c}^t) \wedge \bigvee_{\substack{h \in \text{HINTS} \\ c \in h}} \text{mark}_{x,h}^{t+1} \right)$$

We suppose agent \mathbf{m} has the following information in its factual mutable belief base Σ_f :

$$\{ \text{active}_{\{r,g\}}^t, \text{pos}_{3,(16,16)}^t, \text{pos}_{6,(16,17)}^t, \text{col}_{3,r}^t, \text{col}_{6,r}^t, \neg \Delta_{\mathfrak{h}}^t \text{col}_{3,r}^t \} \cup \bigcup_{h \in \text{HINTS}} \{ \neg \text{mark}_{3,h}^t \} \cup \bigcup_{t' \leq t} \{ \text{now}^{\geq t'} \}$$

Moreover, for every $t'' > t$, we suppose that $\text{now}^{\geq t''} \notin \Sigma = (\Sigma_m \cup \Sigma_c)$. This means that agent \mathbf{m} knows that the current time is exactly t .

Agent \mathbf{m} is at the last step of its turn. It has decide how to mark one card with an active hint. Thus, its action repertoire Op includes all and only actions of type $+_{\mathbf{m}}^{\text{markHint}_{x,h}^t}$. It is straightforward to verify that action $+_{\mathbf{m}}^{\text{markHint}_{3,\{r,g\}}^t}$ of marking card 3 with the hint $\{r, g\}$ is a solution to the action selection problem $\langle \Sigma, Op, \alpha_G \rangle$ so defined. Indeed, according to the available information in agent \mathbf{m} 's belief base Σ , card 3 at position (16, 16) is unmarked and is adjacent to card 6 at position (16, 17), the two cards are both red and the color of card 3 is not known by agent \mathfrak{h} .

5.6 Goals modeling

We introduce several additional definitions. First, we define the following abbreviation for $t \in \text{TIME}$ and $p \in \text{GRID}$:

$$\text{emp}_p^t \stackrel{\text{def}}{=} \bigwedge_{x \in \text{CARDS}} \neg \text{pos}_{x,p}^t$$

emp_p^t reads “the position p is empty”. Moreover:

$$\begin{aligned} OPOS &: TIME \longrightarrow 2^{GRID} \\ t \mapsto OPOS^t &= \{p \in GRID : \neg \text{emp}_p^t \in \Sigma_f \text{ and } \text{now}^{\geq t} \in \Sigma_f\} \end{aligned}$$

is a function returning the set of all occupied positions at time t .

A set of two different cards x and x' , or a non-empty set X of cards, contains “neighboring cards” if these cards are located in neighboring positions. Thus, we defined the following two functions, the former checks whether two cards x and x' are neighbors, while the latter verifies whether a group of 3 or 4 cards ($X \subseteq CARDS : 3 \leq |X| \leq 4$) are neighbors calling the first function recursively:

$$\begin{aligned} \text{nbgCards}_{\{x,x'\}}^t &\stackrel{\text{def}}{=} \bigvee_{\substack{p \in OPOS^t \\ p' \in NEIG(p)}} (\text{pos}_{x,p}^t \wedge \text{pos}_{x',p'}^t) \\ \text{nbgCards}_X^t &\stackrel{\text{def}}{=} \bigvee_{\substack{x,x' \in X \\ x \neq x'}} (\text{nbgCards}_{X \setminus \{x\}}^t \wedge \text{nbgCards}_{\{x,x'\}}^t) \end{aligned}$$

For every non empty $X, X' \subseteq CARDS : X \cap X' = \emptyset$, $\text{hvECN}_{X,X'}^t$ is true iff there are a card in X and a card in X' that have at least one Empty Common Neighbor position. That is:

$$\text{hvECN}_{X,X'}^t \stackrel{\text{def}}{=} \bigvee_{\substack{x \in X \\ p \in OPOS^t}} \bigvee_{\substack{x' \in X' \\ p' \in OPOS^t : \\ NEIG(p) \cap NEIG(p') \neq \emptyset}} (\text{pos}_{x,p}^t \wedge \text{pos}_{x',p'}^t)$$

Finally, for every non empty $X \subseteq CARDS$, hvEN_X^t is true iff X has at least one empty neighbor position :

$$\text{hvEN}_X^t \stackrel{\text{def}}{=} \bigvee_{\substack{x \in X \\ p \in OPOS^t \\ p' \in NEIG(p)}} (\text{pos}_{x,p}^t \wedge \text{emp}_{p'}^t)$$

In the rest of this section, we associate to each action that the agent must perform a set of goals. It is the satisfaction of one of these goals that will cause the agent to perform a specific action. Among these goals, there is always one (called “default goal”) corresponding to a strategy to be executed by default when all the other more specific “specific goals” have failed or have been judged unreachable. A goal is considered “attainable” when its precondition is satisfied before attempting to satisfy this goal. This precondition is there to inform the agent \mathbf{m} that it is not useful to look for a plan to satisfy this goal if this precondition is false. If the precondition is satisfied, this does not mean that the goal will always be satisfied, only that it is possible that it will be.

Of course, the set of specific goals given for each action is not complete, and one can imagine defining other specific goals. Likewise, there are probably other default goals. This is only to give examples and show how to proceed.

5.6.1 Observe actions

Here we describe a set of goals that will guide the agent \mathbf{m} to observe one card rather than another. If the agent fails to successfully execute a specific goal, then it will try to satisfy the default goal of looking at a random card.

(Specific) goal: observe a card around a cards group of the same color.

This goal concerns the observation of a card around a set X of cards of the same color, known by the agent \mathbf{m} . By definition, this means that \mathbf{m} aims there exists a card x' not belonging to X such that: the color of x' is not known by \mathbf{m} at time t , x' is in a neighbour position with X at time t , and the next instant (at $t + 1$) \mathbf{m} will know the color of x' .

A precondition that must be verified before trying to satisfy this goal and that all the cards of the set X are indeed cards located in neighboring positions at time t .

Let $X \subseteq CARDS : 1 \leq |X| \leq 3$, a set of cards of the same color:

$$\begin{aligned} \alpha_{obsArroundCards(X)}^t &\stackrel{\text{def}}{=} \bigvee_{\substack{x' \in CARDS \\ x' \notin X}} \left(\left(\bigwedge_{c \in COLORS} \neg \Delta_{\mathbf{m}} col_{x',c}^t \right) \wedge \right. \\ &\quad \left. \text{nbgCards}_{X \cup \{x'\}}^t \wedge \left(\bigvee_{c \in COLORS} \Delta_{\mathbf{m}} col_{x',c}^{t+1} \right) \right) \\ \mathcal{P}_g(\alpha_{obsArroundCards(X)}^t) &\stackrel{\text{def}}{=} \text{nbgCards}_X^t \end{aligned}$$

(Specific) goal: observe a card whose color is unknown. If agent \mathbf{m} considers that it is not possible to satisfy the previous goal, then \mathbf{m} will aim to observe the color of a card that it does not already know (this is the precondition of the action).

Let $x \in CARDS$:

$$\begin{aligned} \alpha_{obsUnknownColorCard(x)}^t &\stackrel{\text{def}}{=} \bigvee_{c \in COLORS} \Delta_{\mathbf{m}} col_{x,c}^{t+1} \\ \mathcal{P}_g(\alpha_{obsUnknownColorCard(x)}^t) &\stackrel{\text{def}}{=} \bigwedge_{c \in COLORS} \neg \Delta_{\mathbf{m}} col_{x,c}^t \end{aligned}$$

(default) goal: observe a card randomly. When all other goals failed or were impossible to achieve, then agent \mathbf{m} performs the default goal of observing a random card. Because of the previous goal, \mathbf{m} already knows the color of this card, but it is still obliged by the rules of the game to observe the color of a card. This action is always executable (no precondition).

Let $x \in CARDS$:

$$\alpha_{obsRandomly(x)}^t \stackrel{\text{def}}{=} \bigvee_{c \in COLORS} \Delta_m col_{x,c}^{t+1}$$

$$\mathcal{P}_g(\alpha_{obsRandomly(x)}^t) \stackrel{\text{def}}{=} \top$$

How to compute parameters of goals? The parameters of a goal are the free variables in the definition of this goal. For convenience, we define a useful function in Algorithm 2.

Algorithm 2 Useful functions for goals parameters computation

```

1: function GETSUBSETSGTLEQ( $X, \tau_m, \tau_M$ )
2:    $S \leftarrow \emptyset$ 
3:   for all  $X' \in 2^X$  do
4:     if  $\tau_m < |X'| \leq \tau_M$  then
5:       PUSH( $X', S$ )
6:     end if
7:   end for
8:   return  $S$ 
9: end function
10:
11: function GETELEMENTSOFsize( $X, \omega$ )
12:    $S \leftarrow \emptyset$ 
13:   for all  $X' \in X$  do
14:     if  $|X'| == \omega$  then
15:       PUSH( $X', S$ )
16:     end if
17:   end for
18:   return  $S$ 
19: end function

```

The GETSUBSETSGTLEQ function returns all the subsets of the set X whose number of elements is strictly greater than τ_m and is lower or equal to τ_M . For example, GETSUBSETSGTLEQ($\{\{1, 2\}, 0, |\{1, 2\}|\}$) returns $\{\{1\}, \{2\}, \{1, 2\}\}$ (the empty set is not in the answer but $\{1, 2\}$ is). Moreover, the GETELEMENTSOFsize function returns all elements of the input parameter X whose length is exactly ω (X is a set of sets, and $\omega \in \mathbb{N}$).

Goals parameters, the ordering of the goals and the verification of their precondition are determined by the metalogical Algorithm 3.

In this algorithm, the TRYTOOBSERVE function checks, for each element x of S (line 4), that the goal precondition for x is true (line 5), and if so attempts to accomplish the goal for x (line 6). If at least one attempt is successful the function returns true. If for every x the precondition is false or the goal not satisfied, the

Algorithm 3 Computation of the arguments for observe goals

```

1: function TRYTOOBSERVE( $S, \alpha_g^t$ )
2:    $success \leftarrow \text{false}$ 
3:   while  $S \neq \emptyset$  &&  $\neg success$  do
4:      $x \leftarrow \text{POP}(S)$ 
5:     if  $\mathcal{P}_g(\alpha_{g(x)}^t)$  then
6:        $success \leftarrow \text{CHECKPLAN}(\alpha_{g(x)}^t)$ 
7:     end if
8:   end while
9:   return  $success$ 
10: end function
11:
12:  $S \leftarrow \emptyset$ 
13: for all  $c \in \text{COLORS}$  do
14:    $X_c \leftarrow \{x \in \text{CARDS} : \Delta_m \text{col}_{x,c}^t \in \Sigma_f\}$ 
15:    $S \leftarrow S \cup \text{GETSUBSETSGTLEQ}(X_c, 0, |X_c|)$ 
16: end for
17:  $S' \leftarrow \text{SORTBYDECREASINGSIZE}(S)$   $\triangleright \forall X_i, X_{i+1} \in S', |X_i| \geq |X_{i+1}|$ 
18:
19: if  $\neg \text{TRYTOOBSERVE}(S', \alpha_{\text{obsArroundCards}}^t)$  then
20:    $X \leftarrow \{x \in \text{CARDS} : \exists c \in \text{COLORS}, \Delta_m \text{col}_{x,c}^t \in \Sigma_f\}$ 
21:    $X' \leftarrow \text{CARDS} \setminus X$   $\triangleright$  The set of cards whose color is unknown
22:   if  $\neg \text{TRYTOOBSERVE}(X', \alpha_{\text{obsUnknownColorCard}}^t)$  then
23:      $\text{TRYTOOBSERVE}(\text{CARDS}, \alpha_{\text{obsRandomly}}^t)$ 
24:   end if
25: end if

```

function returns false. Note that the function parameter S is a set of elements that can be cards or non empty subsets of cards.

So, in Algorithm 3, we start by computing the set S which contains, for each color (line 13), the set of cards of this color that are known by the agent \mathbf{m} (line 14), as well as all its non-empty subsets (line 15). Finally, S' (line 17) is the set such that: each element is a subset of cards of the same color, and all these elements are sorted by size decreasing. For example, suppose that:

$$\Sigma_f = \{\Delta_m \text{col}_{15,b}^t, \Delta_m \text{col}_{6,b}^t, \Delta_m \text{col}_{1,g}^t, \Delta_m \text{col}_{13,g}^t, \Delta_m \text{col}_{7,g}^t, \Delta_m \text{col}_{9,y}^t\}.$$

So, we have:

$$S = \{\{15, 6\}, \{15\}, \{6\}, \{1, 13, 7\}, \{1, 13\}, \{1, 7\}, \{13, 7\}, \{1\}, \{13\}, \{7\}, \{9\}\}$$

$$S' = \{\{1, 13, 7\}, \{15, 6\}, \{1, 13\}, \{1, 7\}, \{13, 7\}, \{15\}, \{6\}, \{1\}, \{13\}, \{7\}, \{9\}\}$$

The end of Algorithm 3 describes the ordering of the different goals: the agent

m will first try to observe a card around a group of neighboring cards of the same color (line 19), then it will try to look at a random card among those it does not know (line 20 to line 22), then if no previous goal has been achieved, then he will look at a random card (line 23).

5.6.2 Move actions

Goal: group 4 cards of the same color. Let $X \subseteq CARDS$ such that $|X| = 4$ and every cards in X have the same color:

$$\begin{aligned} \alpha_{groupCards_4}^t(X) &\stackrel{\text{def}}{=} \text{nbgCards}_X^{t+1} \\ \mathcal{P}_g(\alpha_{groupCards_4}^t(X)) &\stackrel{\text{def}}{=} \neg \text{nbgCards}_X^t \wedge \\ &\left(\bigvee_{\substack{X' \subseteq X \\ |X'|=3}} (\text{nbgCards}_{X'}^t \wedge \text{hvEN}_{X'}^t) \vee \bigvee_{\substack{X' \subseteq X \\ |X'|=2}} (\text{nbgCards}_{X'}^t \wedge \text{hvECN}_{X \setminus X', X'}^t) \right) \end{aligned}$$

The precondition above reads that there exists a plan to group the 4 cards in X iff: 1) they are currently not neighboring cards; and 2) either a subset X' of 3 cards is a set of neighboring cards and at least one card of X' has a free neighboring position, or a subset X' of 2 cards is a set of neighboring cards and at least one of the other cards of X (not in X') has a neighboring free position in common with a card of X' .

Goal: group 3 cards of the same color. Let $c \in COLORS$, $X \subseteq CARDS$ such that $|X| = 3$ and every cards in X have the same color:

$$\begin{aligned} \alpha_{groupCards_3}^t(X) &\stackrel{\text{def}}{=} \text{nbgCards}_X^{t+1} \\ \mathcal{P}_g(\alpha_{groupCards_3}^t(X), c) &\stackrel{\text{def}}{=} \neg \text{nbgCards}_X^t \wedge \\ &\bigwedge_{x \in CARDS \setminus X} \left(\text{col}_{x,c}^t \rightarrow \neg \text{nbgCards}_{X \cup \{x\}}^t \right) \wedge \\ &\left(\bigvee_{\substack{X' \subseteq X \\ |X'|=2}} (\text{nbgCards}_{X'}^t \wedge \text{hvEN}_{X'}^t) \vee \bigvee_{\substack{x, x' \in X \\ x \neq x'}} \text{hvECN}_{\{x\}, \{x'\}}^t \right) \end{aligned}$$

The precondition above reads that there exists a plan to group the 3 cards in X having the same color c iff: 1) the cards in X are currently not neighboring cards; 2) there does not exist a card of color c not in X forming (together with X) a set of neighboring cards ; and 3) either a subset X' of 2 cards is a set of neighboring cards and at least one card of X' has a free neighboring position, or at least 2 cards in X have a neighboring free position in common.

Goal: group 2 cards of the same color. Let $c \in COLORS$, $X \subseteq CARDS$ such that $|X| = 2$ and every cards in X have the same color:

$$\begin{aligned} \alpha_{groupCards_2(X)}^t &\stackrel{\text{def}}{=} \text{nbgCards}_X^{t+1} \\ \mathcal{P}_g(\alpha_{groupCards_2(X)}^t, c) &\stackrel{\text{def}}{=} \neg \text{nbgCards}_X^t \wedge \\ &\bigwedge_{x \in CARDS \setminus X} \left(col_{x,c}^t \rightarrow \bigwedge_{x' \in X} \neg \text{nbgCards}_{\{x,x'\}}^t \right) \wedge \bigvee_{x \in X} \text{hvEN}_{\{x\}}^t \end{aligned}$$

The precondition above reads that there exists a plan to group 2 cards x and x' having the same color c iff: 1) the cards in X are currently not neighboring cards; 2) there does not exist a card of color c not in X forming, together with a card in X , a set of neighboring cards ; and 3) at least 1 card in X has a neighboring free position in common.

Goal: move randomly a card. Let $x \in CARDS$:

$$\begin{aligned} \alpha_{randomMove(x)}^t &\stackrel{\text{def}}{=} \bigvee_{\substack{p \in OPOS^t \\ p' \in GRID \setminus OPOS^t}} (pos_{x,p}^t \wedge pos_{x,p'}^{t+1}) \\ \mathcal{P}_g(\alpha_{randomMove(x)}^t) &\stackrel{\text{def}}{=} \left(\bigwedge_{c \in COLORS} \neg \Delta_{\mathbf{m}} col_{x,c}^t \right) \vee \\ &\bigvee_{c \in COLORS} \left(col_{x,c}^t \wedge \bigwedge_{\substack{x' \in CARDS \\ x' \neq x}} (col_{x',c}^t \rightarrow \neg \text{nbgCards}_{\{x,x'\}}^t) \right) \end{aligned}$$

How to compute arguments of goals? The parameters of goals and their precondition are computed directly from the mutable belief base Σ_m of agent \mathbf{m} .

In Algorithm 4, the function GETMARKEDCARDS has a parameter X (a set of cards having a same color). If X does not already contain 4 cards (line 4), then this function will return the set of all cards x such that: 1) there exist a hint h that includes the color c of cards in X (line 5); 2) x is marked by h (line 6) and 3) \mathbf{m} believes that x does not have a color not represented in h (that is, \mathbf{m} has no reason to believe that h gives a false indication about the color of x). Note that H_c is computed only if the number of known cards is lower than 4 (line 4) because when all the cards of a same color are known, we do not try to generate moves near cards with a hint.

The function TRYTOMOVE has three parameters: a set S of elements (either subsets of cards, or cards), a given goal α_g^t and a boolean b . The only difference of this function from the TRYTOOBSERVE function is the case where b is true (line 16): in this case, an additional parameter (the color of cards in X_c) is used to check the goal precondition. This additional parameter is required when α_g^t is either $\alpha_{groupCards_3}^t$ (to group 3 cards) or $\alpha_{groupCards_2}^t$ (to group 2 cards), which indicates a true value for b .

Algorithm 4 Computation of the arguments for move goals

```

1: function GETMARKEDCARDS( $X$ )
2:                                     ▷  $X$  is a set of cards of the same color
3:    $c \leftarrow$  GETCOLOR( $X$ )           ▷ return the color of cards in  $X$ 
4:   if  $|X| < 4$  then
5:      $H_c \leftarrow \{x \in CARDS : \exists h \in HINTS \text{ such that } c \in h \text{ and}$ 
6:        $\Sigma_m \cup \Sigma_c \models \Box_m (mark_{x,h}^t \wedge \bigwedge_{c' \in COLORS \setminus h} \neg col_{x,c'}^t)\}$ 
7:   else
8:      $H_c \leftarrow \emptyset$ 
9:   end if
10: end function
11:
12: function TRYTOMOVE( $S, \alpha_g^t, b$ )
13:    $success \leftarrow$  false
14:   while  $S \neq \emptyset$  &&  $\neg success$  do
15:      $x \leftarrow$  POP( $S$ )
16:     if  $b$  then
17:        $Precond \leftarrow \mathcal{P}_g(\alpha_{g(x)}^t, \text{GETCOLOR}(x))$ 
18:     else
19:        $Precond \leftarrow \mathcal{P}_g(\alpha_{g(x)}^t)$ 
20:     end if
21:     if  $b$  then
22:        $success \leftarrow$  CHECKPLAN( $\alpha_{g(x)}^t$ )
23:     end if
24:   end while
25:   return  $success$ 
26: end function
27:
28:  $S \leftarrow \emptyset$ 
29: for  $c \in COLORS$  do
30:    $X_c \leftarrow \{x \in CARDS : \Delta_m col_{x,c}^t \in \Sigma_m\}$ 
31:    $H_c \leftarrow$  GETMARKEDCARDS( $X_c$ )
32:    $S \leftarrow$  GETSUBSETSGTLEQ( $X_c \cup H_c, 1, 4$ )
33: end for
34:  $S' \leftarrow$  SORTBYDECREASINGSIZEHINTS( $S$ )           ▷ See below for explanations
35:
36:  $X_4 \leftarrow$  GETELEMENTSOFSIZE( $S', 4$ )
37: if  $\neg$ TRYTOMOVE( $X_4, \alpha_{groupCards_4}^t, \text{false}$ ) then
38:    $X_3 \leftarrow$  GETELEMENTSOFSIZE( $S', 3$ )
39:   if  $\neg$ TRYTOMOVE( $X_3, \alpha_{groupCards_3}^t, \text{true}$ ) then
40:      $X_2 \leftarrow$  GETELEMENTSOFSIZE( $S', 2$ )
41:     if  $\neg$ TRYTOMOVE( $X_2, \alpha_{groupCards_2}^t, \text{true}$ ) then
42:       TRYTOMOVE( $CARDS, \alpha_{randomMove}^t, \text{false}$ )
43:     end if
44:   end if
45: end if

```

Finally, for each color c (line 29) we compute: X_c (line 30) that is the set of cards of color c which are known by \mathbf{m} ; H_c (line 31) that is the set of cards unknown by agent \mathbf{m} but marked by a hint containing color c . Then, we generate the set of subsets of cards (line 32) for which either the color is c or the color is unknown but these cards are marked with a hint containing color c , and we only retain the subsets having between 2 and 4 cards. As it is preferable to first try to group together the greatest possible number of cards of the same color, and preferably cards of which we know the color (that is, not marked by a hint), we sort all the subsets of cards using the **sort** function (line 34). So, $\text{SORTBYDECREASINGSIZEHINTS}(S) = \{X_1, X_2, \dots, X_n\}$ is a set of subsets of cards such that, for every $i \in [1..n-1]$: 1) $|X_i| \geq |X_{i+1}|$; 2) if $|X_i| = |X_{i+1}|$ then $|\mathbf{hints}^t(X_i)| \geq |\mathbf{hints}^t(X_{i+1})|$, where $\mathbf{hints}^t(X) = \{x \in X : \exists h \in \mathbf{HINTS}, \text{mark}_{x,h}^t \in \Sigma_f\}$ is the set of cards in X marked by a hint.

In the rest of Algorithm 4, for each subset of S' we try to group sets of cards of decreasing size (line 36 to line 41). If no plan is successful, then the default goal (move a card randomly, see line 42) is executed (which is always executable).

5.6.3 Mark actions and active actions

Goal: mark a card unknown by \mathfrak{h} with the smallest hint available. Let $x \in \mathbf{CARDS}$ be a cards that is not marked at time t and $h \in \mathbf{HINTS}$ a hint that is active at time t :

$$\begin{aligned} \alpha_{\text{markUnknownCard}(x,h)}^t &\stackrel{\text{def}}{=} \text{mark}_{x,h}^{t+1} \\ \mathcal{P}_g(\alpha_{\text{markUnknownCard}(x,h)}^t) &\stackrel{\text{def}}{=} \bigwedge_{c \in \mathbf{COLORS}} \neg \Delta_{\mathfrak{h}}^t \text{col}_{x,c}^t \wedge \bigvee_{c' \in h} \Delta_{\mathbf{m}} \text{col}_{x,c'}^t \end{aligned}$$

Agent \mathbf{m} has a goal to mark with a hint h a card x for which the color is unknown by agent \mathfrak{h} at time t if and only if x is marked by h at time $t+1$. Note it is not necessary to check if x is not marked at time t because we already suppose it is the case for the x goal parameter.

The precondition ensures that \mathfrak{h} does not know the color of card x yet at time t , and that there exists a color c' in the hint h such that \mathbf{m} believes that x has the color c' .

Goal: mark randomly a card. Let $x \in \mathbf{CARDS}$ be a cards that is not marked at time t and $h \in \mathbf{HINTS}$ a hint that is active at time t :

$$\begin{aligned} \alpha_{\text{markRandomCard}(x,h)}^t &\stackrel{\text{def}}{=} \text{mark}_{x,h}^{t+1} \\ \mathcal{P}_g(\alpha_{\text{markRandomCard}(x,h)}^t) &\stackrel{\text{def}}{=} \bigvee_{c' \in h} \Delta_{\mathbf{m}} \text{col}_{x,c'}^t \end{aligned}$$

The goal is less restrictive than the previous one, since the fact that agent \mathfrak{h} does not know the color of the card that agent \mathbf{m} is trying to mark is not a precondition for trying to reach this goal. Thus, this goal allows to play in the case where at least

one hint is active but its use will not be optimally informative for \mathfrak{h} (who already knows the color of the marked card).

The precondition just ensures there exists a color c' in the hint h such that \mathfrak{m} believes that x has the color c' .

How to compute arguments of goals? The parameters of goals and their precondition are computed directly from the mutable beliefs base Σ_m of agent \mathfrak{m} following Algorithm 5.

Algorithm 5 Computation of the arguments for mark goals

```

1: function TRYTOMARK( $X, H, \alpha_g^t$ )
2:    $success \leftarrow \text{false}$ 
3:   while  $H \neq \emptyset$  &&  $\neg success$  do
4:      $h \leftarrow \text{POP}(H)$ 
5:      $S_X \leftarrow X$ 
6:     while  $S_X \neq \emptyset$  &&  $\neg success$  do
7:        $x \leftarrow \text{POP}(S_X)$ 
8:       if  $\mathcal{P}_g(\alpha_g^t(x, h))$  then
9:          $success \leftarrow \text{CHECKPLAN}(\alpha_g^t(x, h))$ 
10:      end if
11:    end while
12:  end while
13:  return  $success$ 
14: end function
15:
16:  $X \leftarrow \left\{ x \in \text{CARDS} : \Sigma_m \cup \Sigma_c \models \bigvee_{t \in \text{TIME}} (\text{now}^t \wedge \Box_m \bigwedge_{h \in \text{HINTS}} \neg \text{mark}_{x, h}^t) \right\}$ 
17:  $H \leftarrow \left\{ h \in \text{HINTS} : \Sigma_m \cup \Sigma_c \models \bigvee_{t \in \text{TIME}} (\text{now}^t \wedge \Box_m \text{active}_h^t) \right\}$ 
18:  $H_I \leftarrow \text{SORTBYINCREASINGSIZE}(H)$ 
19: if  $\neg \text{TRYTOMARK}(X, H_I, \alpha_{\text{markUnknownCard}}^t)$  then
20:    $H_D \leftarrow \text{SORTBYDECREASINGSIZE}(H)$ 
21:   if  $\neg \text{TRYTOMARK}(X, H_D, \alpha_{\text{markRandomCard}}^t)$  then
22:      $h \leftarrow \text{RANDOM}(HINTS \setminus H)$  ▷ a non active hint
23:     ACTIVE( $h$ )
24:   end if
25: end if

```

In Algorithm 5, we first define the TRYTOMARK function which is similar to TRYTOMOVE and TRYTOOBSERVE previous functions. This function try, for each hint in parameter H (line 3) and each card in parameter X (line 6), to check precondition of the goal in parameter α_g^t (line 8). If this precondition is satisfied, the algorithm try to satisfy the goal itself (line 9). As soon as a goal is satisfied (a plan has been found) the true value is returned, else false is returned.

So, x contains the set of cards that are not marked at time t (line 16), and H contains the set of active hints at time t (line 17).

So, we compute H_I (line 18) that gets elements of H sorted following increasing size (that is, for every $h_j, h_{j+1} \in H_I, |h_j| \leq |h_{j+1}|$), and H_D (line 20) that gets elements of H sorted following decreasing size (that is, for every $h_j, h_{j+1} \in H_D, |h_j| \geq |h_{j+1}|$).

In the rest of Algorithm 5, we try first to satisfy the goal “to mark an unknown card x with an active hint h ”. If it is not possible to satisfy this goal or its preconditions, or if there is no active clue, or if there is no card whose color is unknown to the agent \mathfrak{h} (line 19), then the agent \mathfrak{m} will try to satisfy the next goal “to mark a randomly chosen card x with an active hint h ” (line 21).

Note that when \mathfrak{m} seeks to satisfy the first goal, the set of available indices are classified according to increasing order with respect to their size (i.e., with respect to the number of colors that defines them). This is because the smaller the size of a hint, the more informative that hint is. (A hint of size 1 actually tells agent \mathfrak{h} the real color of the card.) Conversely, when \mathfrak{m} tries to satisfy the second goal, we use the list of hints classified according to their decreasing size. In other words, we start by trying to satisfy the goal with the largest hint (therefore, the least informative hint for agent \mathfrak{h}). This is simply because, in this case, there is no card whose color \mathfrak{h} does not know, so we try to use the least informative hint first (in order to keep the most informative hints for the end of the game).

If no goal is satisfiable, then the agent \mathfrak{m} activates a hint randomly chosen among the set of indices which are not yet active.

Note that in the game, the following situation is possible: agent \mathfrak{m} does not know any green card, but the only remaining hint has already been enabled and its color is blue. In this case, the previous algorithm allow \mathfrak{m} to not execute any action related to hint (\mathfrak{m} passes its turn).

5.7 Implementation and experiments

Implementation. The architecture presented in Figure 6 (See Appendix B for more details about the architecture) works as an integrated system. All the processes, interfaces and exchange of data between the modules work according to the game rules.

The GUI module allows players to perform observations, movements, and active or mark a card with a hint. The activation of hints is randomly generated correctly and according to the specifications for hints detailed in the paragraph “hints” page 74. In addition, the generation of the set of data after each game exactly reflects the current state of the game (current position, marked cards, occupied positions, perimeter of the set of cards, etc.).

Some data can be computed inside the logical model (for example, a position in the grid is occupied if there is a card occupying this position). But the fact of having to recompute them each time the solver is requested, may be very costly in terms of computation time and memory space. Thus, the graphical interface manages a certain amount of data natively (position of the cards, movement authorized or not,

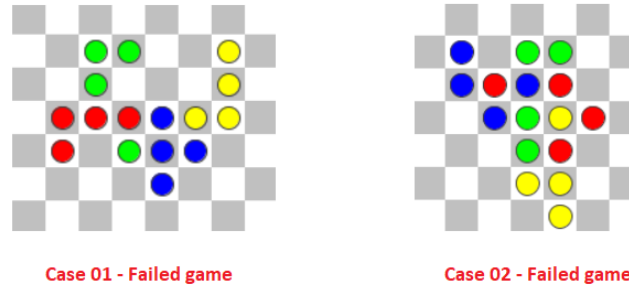


Figure 5.2: Two different scenarios when players lost the game

etc.), and it can be categorized as a *third agent*, which controls, for example, the real colors of the cards as well as the hints generation.

Here is important to mention that the GUI will verify when the players manage to group the cards, because it has the knowledge about the distribution of all cards in the grid.

The belief revision module can drop the older beliefs that contradict new inputs. Similarly, the hierarchy of goals works fine and manages to group the cards according to the cards known by agent *m*.

Data set. Our experiment was constructed as follows. We conducted 10 *experiment sets*. Each experiment set consisted of 20 *games*: 10 games were played by the human-machine *configuration type* (**h-m**) and 10 were played by the human-human configuration type (**h-h**). Finally, the experiment includes 200 games of Yōkai (10 times 20 games). Every time a new game started, the system generated a random distribution of the 16 cards in the grid.

Collaboration level analysis. First, we are interested in measuring the *level of collaboration* between players, regardless of whether players win or fail the game. We propose to calculate the level of collaboration based on the number of *y-groups* (Yōkai groups) when the game is over. We call “y-group” a group of cards of the same color without separation (see p. 91) for this group. If *k* is the size of a y-group (that is, it contains *k* cards), we write it is a *k-y-group*. Then, the *collaboration level* is *n* when, at the end of the game, players have built *n* 4-y-groups.

For instance, consider the example shown in Figure 5.2 above. In this figure, although in both cases the players lost the game, it is easy to see that the collaboration between players was more efficient in **case 1** than in **case 2**: as in **case 1**, we observe three 4-y-groups while in **case 2** we do not observe any 4-y-group, then the collaboration level is 3 in **case 1** and 0 in **case 2**.

As players win the game when they have built four 4-y-groups, they win when the collaboration level is maximal (that is, equal to 4).

In the following, we seek to measure the collaboration level in different cases.

We show in Table 5.1 the results achieved during the games performed in the

first experiment set. Note that the maximal score for a game configuration in an experiment set is 40 (that is, 10 times 4 points).

Experiment set	Game	h-m		h-h	
		# 4-y-groups	Score	# 4-y-groups	Score
1	1	1	-	1	-
	2	1	-	2	-
	3	4	8	2	-
	4	1	-	2	-
	5	3	-	1	-
	6	4	8	4	9
	7	4	7	3	-
	8	2	-	2	-
	9	1	-	1	-
	10	1	-	0	-
Total:		22		18	

Table 5.1: Games performed during experiment set 1

The entire data set of the experiment is presented in Table 5.2 and is plotted in Figure 5.3. In Table 5.2, the average score of the h-m configuration is 26.7, and that of the h-h configuration is 21.6.

Config. type	Experiment set									
	1	2	3	4	5	6	7	8	9	10
h-m	22	28	33	32	28	30	19	26	23	26
h-h	18	23	21	23	24	25	17	19	21	25

Table 5.2: Average total points by experiment set

In Figure 5.3, we can see that the plot representing the h-m configuration (in blue) is always located above that representing the h-h configuration (in red in the figure). This means that the level of collaboration is always higher in the first type than in the second type. (Of course, one can object that a very high level of collaboration can still lead to a low result, but remember that here we define the level of collaboration according to the number of 4-y-groups which is an indicator based on the results of the games.)

Experiment set	h-m		h-h	
	Games won	Best score	Games won	Best score
1	3	8	1	9
2	3	9	2	7
3	5	9	2	8
4	4	11	1	7
5	2	12	1	5
6	4	7	3	8
7	2	8	0	0
8	2	7	1	7
9	1	7	0	0
10	2	8	2	9

Table 5.3: Games won and best scores for each experiment set

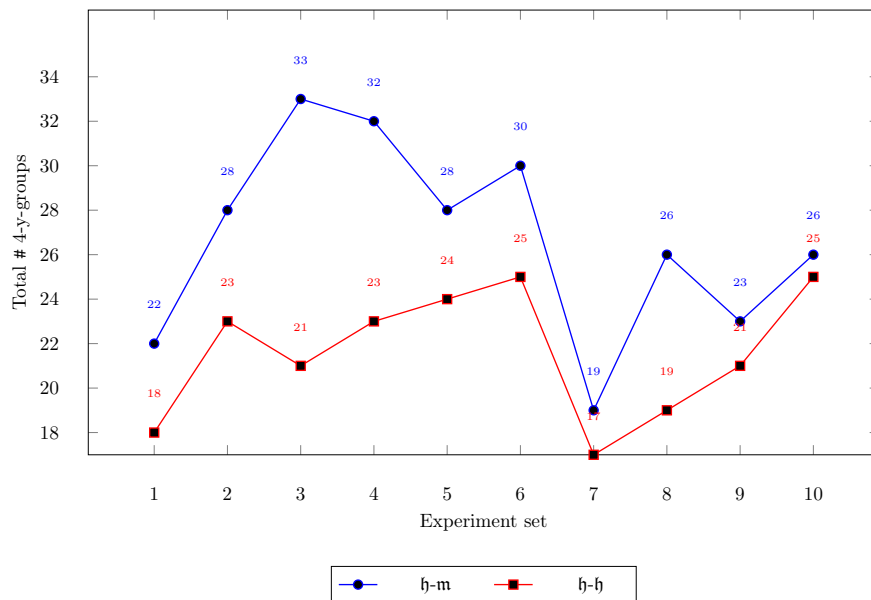


Figure 5.3: Total # 4-y-groups by experiment set

Won games and scores analysis. The next step was to count the number of winning games between the ten games performed by type of configuration in each experiment set. Even more, we are interested in the highest score achieved between the ten games performed by type of configuration. Recall the score is computed according to the formula detailed in Section 5.2. We represented both measures in Table 5.3.

We show in Figure 5.4 the number of games won for each type of configuration. Note that the best possible result by set is 10. Again, we can see that the number of games won in blue (h-m) is always greater than or equal to that in red (h-h). The best result is 5 (that is, one out of two games was won within this set) and is

obtained by $\mathfrak{h}\text{-m}$ collaboration. For $\mathfrak{h}\text{-h}$ collaboration, the best score is only 3.

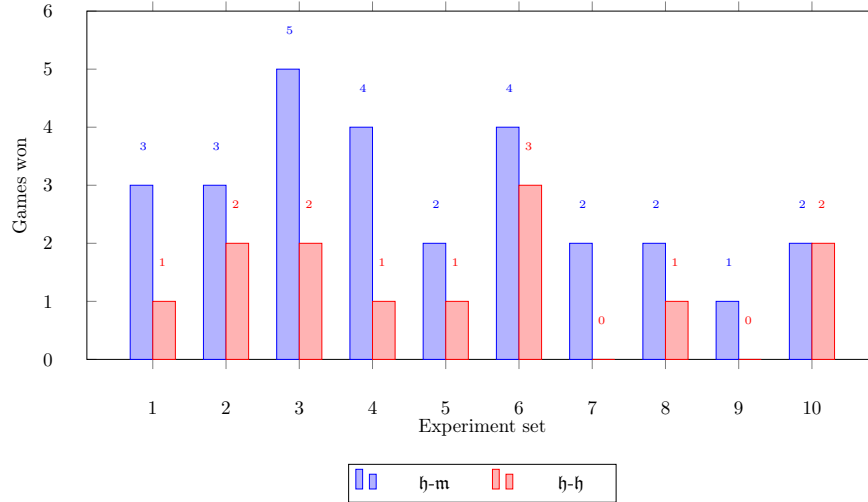


Figure 5.4: Number of games won in each experimental set by each game configuration

In Figure 5.5 we show the best score achieved in each experiment set and for each type of collaboration. We can observe that in three experiment sets the $\mathfrak{h}\text{-m}$ configuration was able to overcome the score achieved by the $\mathfrak{h}\text{-h}$ configuration.

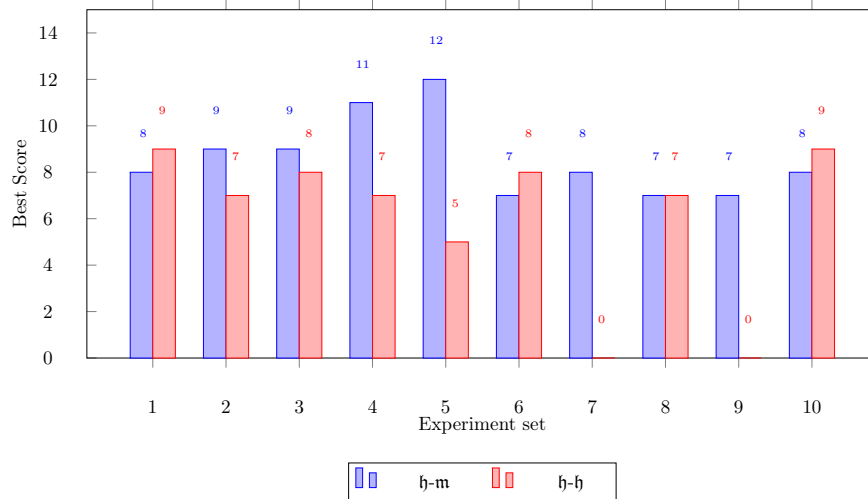


Figure 5.5: Best score achieved among games performed in each experimental set by each game configuration

5.8 Discussion

Let's first compare the level of collaboration according to the type of game configuration. We can see in Figure 5.3 that the $\mathfrak{h}\text{-}\mathfrak{m}$ collaboration overcomes the $\mathfrak{h}\text{-}\mathfrak{h}$ collaboration in grouping cards of the same color. Among the 10 experiment sets, the $\mathfrak{h}\text{-}\mathfrak{h}$ configuration was close to reaching the same level of collaboration as the $\mathfrak{h}\text{-}\mathfrak{m}$ configuration in only two sets (7 and 9) .

Additionally, analysis of the scoring system from the Figure 5.4 shows that the number of games won in the configuration $\mathfrak{h}\text{-}\mathfrak{m}$ is higher than that of the configuration $\mathfrak{h}\text{-}\mathfrak{h}$ in all experiment sets. Moreover, the Figure 5.5 shows that the highest score obtained in the configuration $\mathfrak{h}\text{-}\mathfrak{m}$ is on average better than that in the configuration $\mathfrak{h}\text{-}\mathfrak{h}$, since there are only three cases (sets 1, 6 and 10) where the configuration $\mathfrak{h}\text{-}\mathfrak{h}$ exceeds the best score achieved in the configuration $\mathfrak{h}\text{-}\mathfrak{m}$.

By analyzing the games played during the experiment, it seems that if the hints activated at the start of the game are those with a single color, then there is a greater probability of winning the game because one player can use these hints to guide the other player about the cards it knows and the other doesn't. Similarly, marking cards with a hint having three colors sometimes makes it possible to deduce the color of a card according to the other active hints. For example, suppose agent \mathfrak{h} observes a card while there are three active hints $\{y, b\}$, $\{y, r\}$ and $\{y, g, r\}$. Suppose also that a player marks a card with the hint $\{y, g, r\}$. In such a case, and assuming that we keep in mind that the smaller hints are played first because they are more informative, we could deduce that the marked card is green. In order not to complicate our model, this type of reasoning has not been taken into account for the moment. Nevertheless, it is not a technical impossibility: as we did for the rules of the type "if a card is marked with a hint h then its color cannot be a color not belonging to h " (see page 87), we could integrate similar rules into the Σ_h set of revisable beliefs of the agent \mathfrak{m} so that we can remove them if necessary. For example, suppose that using such a rule, the agent \mathfrak{m} deduces that the marked card is green, and suppose that later, by observing other cards, \mathfrak{m} actually knows the 4 cards that are really green. \mathfrak{m} should no longer deduce from the clue that the marked card is green and \mathfrak{m} just has to revise its beliefs by removing the rule allowing it to deduce the color of the marked card.

Following the Yōkai rules, we can only activate a hint or use it for marking a card. Currently, the algorithm that manages the fact of marking a card using a hint (see Algorithm 5) only allows activating a new hint if no more hints are playable or available. Nevertheless, sometimes it is better to activate a new hint than to use a hint in a non-informative way for the other player. For example, this case can occur if the only active hint is $\{g\}$ and I know that the other player already knows all the green cards that I know. The algorithm can be improved, although the idea was not in this work to describe all the possible goals and the best strategy to seek to satisfy them, but rather to simply show how to proceed, and the capacity of the formal language to do this work.

5.9 Conclusion

We introduced a simple epistemic language to represent the knowledge and actions of an artificial player in the context of the cooperative board game Yōkai. We have shown that this game requires a combination of theory of mind (ToM) and temporal and spatial reasoning to be played effectively by an artificial agent. Our approach relies on SAT given the existence of a translation preserving the polysize satisfiability of the epistemic language into propositional logic.

We implemented the game following the same architecture used for the implementation of the artificial assistant detailed in Chapter 4. We noticed that the two main factors that facilitated the implementation were the belief update process and the delegation of some validations to the GUI.

We found that the dynamic generation of actions improved the planner's performance. This improvement is based on the fact that at each time point the machine performs only one action, which type is known in advance based on its sequence in the round. For instance, if the current action is in fact the machine's third action, then it is of type moving a card. Thus, it is not efficient to perform planning with repertoire of actions that considers all the possible kinds of operations. Thus, if the action the machine has to perform is to move a card, then the system only generates moving actions in the repertoire of actions. Furthermore, the target positions will be around the perimeter of the cards in the grid, avoiding separating the cards into two isolated groups. We demonstrated that the hierarchy of goals was the appropriate technique to guide the machine's actions during the game.

Future work could be organized in two steps. First, we could try to implement a machine-machine version of the Yōkai game and compare the effectiveness of this collaboration with our results in the $\mathfrak{h}\text{-m}$ and $\mathfrak{h}\text{-h}$ configuration. Second, we could include a machine learning module in the system architecture, which will endow the machine agent with the necessary skills to learn new strategies based on previous games. Finally, just as we did with our basic GUI chatbot in Chapter 4, we could detach our java interface and replace it with a web version that will allow us to make the game available to a wider audience.

Conclusion

In concluding this thesis, we give a summary of our work. How did we answer our research questions? Which new concepts did we introduce? Which results did we show? Which open questions remain?

Summary

We started our investigations with an overview of the standard frameworks for Epistemic Logic, Doxastic Logic, Dynamic Epistemic Logic, Logic of Doxastic Attitudes and Epistemic Planning. Finally, we studied SAT and QBF in Chapter 1. The chapters which followed tackled four research questions:

1. Can we build a logical framework to represent and reason about an agent's beliefs, desires, and intentions and whose satisfiability problem could be reduced to SAT?
2. Can we propose an integrated architecture for cognitive planning in the context of our logical framework? Moreover, is it possible to find an efficient encoding for our cognitive planning problem?
3. How can we implement our integrated architecture for cognitive planning using functional programming?
4. How to apply our framework for cognitive planning to model and implement a cooperative game that uses ToM as well as temporal and spatial reasoning?

In Chapter 2 we achieved our goal of finding an NP-fragment of the LDA logic by avoiding the recursive call to the implicit belief modality. We proved that the satisfiability problem of our NP-fragment is reduced to SAT. Furthermore, we provided a set of reduction axioms to transform formulas expressed in our NP-fragment to their propositional logic version with sub-indexes to represent the Kripke worlds and special atoms to simulate accessibility relations. Therefore, our first research question can be answered with a clear yes.

In Chapter 3, we answered our second question by proposing an integrated framework for cognitive planning. We presented a general architecture that considers two core modules: belief revision and cognitive planning module. Next, we defined two types of planning problem: the informative planning problem and the interrogative planning problem. Two examples are proposed to illustrate both approaches. The first example describes a scenario where an artificial assistant aims to induce in the human agent a potential intention to choose the ideal sport recommended for him, based on the user's preferences. In the second example, an artificial agent plays the role of a virtual coach, trying to persuade the human agent to practice physical

activity using the principles of motivational interviewing. The virtual coach tries to reach her goal by making the human agent aware of the inconsistency between his current behavior (not doing physical exercise) and his desires (to be in good health). We formalized the belief revision functionality, where the belief base is divided into core beliefs and mutable beliefs. At the end of this chapter, we introduced an optimal QBF encoding for an informative planning problem.

In Chapter 4 we tackled our third research question by moving from mathematics to programming. We implemented our model for cognitive planning in the functional programming paradigm using Ocaml. We implemented the set of translations proposed in Chapter 2 in an independent module called *translator* integrated with the belief revision and cognitive planning modules represented in the system architecture detailed in Chapter 3. The cognitive planning module generates the planning formulas using an initial state, a set of actions and a persuasive goal. Thereafter, the planning module calls the translator and once the planning formula is transformed into propositional logic, the cognitive planning module calls the TouIST solver to verify the validity of the formula. An additional technical point here was the introduction of parallelism to improve the system's performance. In fact, in our implementation of the scenario presented in example 1 (Section 3.6), in which an artificial assistant aims to persuade the human agent to practice a sport based on her preferences, the system starts a set of threads per each possible ideal sport. When one of the threads is able to find a valid plan, the system saves it as an abstract plan. Later, the GUI translates the abstract plan into a natural language expression and shows it to the human agent.

A second application for our cognitive planning framework was a cooperative board game called Yōkai. In Chapter 5, we presented the logical modeling for Yōkai in which in order to be played effectively, participants are required to reason about time, space as well as apply principles of ToM. We first presented a two-agent timed variant of the language and the semantics of our logical framework presented in Chapter 2. Second, we modeled the game considering the static (the different rules, the initial state of the game, etc.) and dynamic aspects (the effects of the actions during the evolution of the game and its preconditions). In terms of our two previous definitions, it seems that to make it fit into a planning problem we are missing the goals. In Section 5.6, the goals were modeled for each of the actions type. One of the main challenges was implementing the functionality to allow the machine to select the best strategy depending on the action type. We demonstrated that the hierarchy of goals was the correct technique for guiding the artificial agent's actions during the game.

Appendices

Appendix A

Detailed proof of Theorem 10

In this section, we are going to provide a polysize reduction of the satisfiability problem of \mathcal{L} to SAT. The reduction consists of three steps.

As a first step, we put \mathcal{L} formulas in negation normal form (NNF) via the following function nnf :

$$\begin{aligned}
nnf(p^t) &= p^t, \\
nnf(now^{\geq t}) &= now^{\geq t}, \\
nnf(\Delta_h^t \alpha) &= \Delta_h^t \alpha, \\
nnf(\Delta_m \alpha) &= \Delta_m \alpha, \\
nnf(\Box_m \alpha) &= \Box_m nnf(\alpha), \\
nnf(\Diamond_m \alpha) &= \Diamond_m nnf(\alpha), \\
nnf(\varphi \wedge \psi) &= nnf(\varphi) \wedge nnf(\psi), \\
nnf(\varphi \vee \psi) &= nnf(\varphi) \vee nnf(\psi), \\
nnf(\neg p^t) &= \neg p^t, \\
nnf(\neg now^{\geq t}) &= \neg now^{\geq t}, \\
nnf(\neg \Delta_h^t \alpha) &= \neg \Delta_h^t \alpha, \\
nnf(\neg \Delta_m \alpha) &= \neg \Delta_m \alpha, \\
nnf(\neg \neg \varphi) &= nnf(\varphi), \\
nnf(\neg(\varphi \wedge \psi)) &= nnf(\neg \varphi \vee \neg \psi), \\
nnf(\neg(\varphi \vee \psi)) &= nnf(\neg \varphi \wedge \neg \psi), \\
nnf(\neg \Box_m \alpha) &= \Diamond_m nnf(\neg \alpha), \\
nnf(\neg \Diamond_m \alpha) &= \Box_m nnf(\neg \alpha).
\end{aligned}$$

Let us define the NNF variant \mathcal{L}_0^{NNF} of the language \mathcal{L}_0 by the following grammar:

$$\begin{aligned}
\beta ::= & p^t \mid \neg p^t \mid now^{\geq t} \mid \neg now^{\geq t} \mid \Delta_h^t \alpha \mid \neg \Delta_h^t \alpha \mid \Delta_m \alpha \mid \\
& \neg \Delta_m \alpha \mid \beta_1 \wedge \beta_2 \mid \beta_1 \vee \beta_2
\end{aligned}$$

where p^t ranges over Atm , t ranges over $TIME$ and α ranges over \mathcal{L}_0 .

Furthermore, let us define the language \mathcal{L}^{NNF} by the following grammar. For β ranging over \mathcal{L}_0^{NNF} :

$$\varphi ::= \beta \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_m \beta \mid \Diamond_m \beta.$$

Proposition 6 *Let $\varphi \in \mathcal{L}$. Then, $\varphi \leftrightarrow nnf(\varphi)$ is valid in the class \mathbf{M} , and $nnf(\varphi) \in \mathcal{L}^{NNF}$.*

Note that the size of $nnf(\varphi)$ is polynomial in the size of φ .

As a second step, we define the following modal language \mathcal{L}^{Mod} into which the language \mathcal{L}^{NNF} will be translated:

$$\begin{aligned}\omega & ::= q \mid \neg\omega \mid \omega_1 \wedge \omega_2 \mid \omega_1 \vee \omega_2, \\ \varphi & ::= q \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \blacksquare\omega \mid \blacklozenge\omega\end{aligned}$$

where q ranges over the following set of atomic formulas:

$$\begin{aligned}Atm^+ = Atm \cup \{p_{now \geq t} : t \in TIME\} \cup \{p_{\Delta_m \alpha} : \alpha \in \mathcal{L}_0\} \cup \\ \{p_{\Delta_b^t \alpha} : t \in TIME \text{ and } \alpha \in \mathcal{L}_0\}.\end{aligned}$$

So $p_{now \geq t}$, $p_{\Delta_m \alpha}$ and $p_{\Delta_b^t \alpha}$ are nothing but special propositional variables.

We interpret the language \mathcal{L}^{Mod} w.r.t. a pair (M, w) , called pointed Kripke model, where $M = (W, \Rightarrow, \pi)$, W is a set of worlds, $\Rightarrow \subseteq W \times W$ and $\pi : Atm^+ \rightarrow 2^W$. (Boolean cases are again omitted as they are defined in the usual way.)

Definition 29 *The semantic interpretation for formulas in \mathcal{L}^{Mod} w.r.t. a pointed Kripke model (M, w) is as follows:*

$$\begin{aligned}(M, w) \models q & \iff w \in \pi(q); \\ (M, w) \models \blacksquare\omega & \iff \forall v \in W \text{ if } w \Rightarrow v \text{ then } (M, v) \models \omega; \\ (M, w) \models \blacklozenge\omega & \iff \exists v \in W \text{ s.t. } w \Rightarrow v \text{ and } (M, v) \models \omega.\end{aligned}$$

The class of pointed Kripke models is denoted by \mathbf{K} . Satisfiability and validity of formulas in \mathcal{L}^{Mod} relative to the class \mathbf{K} is defined in the usual way.

Let $tr_0 : \mathcal{L}_0 \rightarrow \mathcal{L}^{Mod}$ be a translation such that:

$$\begin{aligned}tr_0(p^t) & = p^t, \\ tr_0(\neg\alpha) & = \neg tr_0(\alpha), \\ tr_0(\alpha_1 \wedge \alpha_2) & = tr_0(\alpha_1) \wedge tr_0(\alpha_2), \\ tr_0(\Delta_i \alpha) & = p_{\Delta_i \alpha}\end{aligned}$$

Let $tr_1 : \mathcal{L}^{NNF} \rightarrow \mathcal{L}^{Mod}$ be a translation such that:

$$\begin{aligned}
tr_1(p^t) &= p^t, \\
tr_1(\neg p^t) &= \neg p^t, \\
tr_1(now^{\geq t}) &= p_{now^{\geq t}}, \\
tr_1(\neg now^{\geq t}) &= \neg p_{now^{\geq t}}, \\
tr_1(\Delta_{\mathfrak{h}}^t \alpha) &= p_{\Delta_{\mathfrak{h}}^t \alpha}, \\
tr_1(\neg \Delta_{\mathfrak{h}}^t \alpha) &= \neg p_{\Delta_{\mathfrak{h}}^t \alpha}, \\
tr_1(\varphi_1 \wedge \varphi_2) &= tr_1(\varphi_1) \wedge tr_1(\varphi_2), \\
tr_1(\varphi_1 \vee \varphi_2) &= tr_1(\varphi_1) \vee tr_1(\varphi_2), \\
tr_1(\Delta_{\mathfrak{m}} \alpha) &= p_{\Delta_{\mathfrak{m}} \alpha} \wedge \blacksquare tr_0(\alpha), \\
tr_1(\neg \Delta_{\mathfrak{m}} \alpha) &= \neg p_{\Delta_{\mathfrak{m}} \alpha}, \\
tr_1(\Box_{\mathfrak{m}} \beta) &= \blacksquare tr_0(\beta), \\
tr_1(\Diamond_{\mathfrak{m}} \beta) &= \blacklozenge tr_0(\beta).
\end{aligned}$$

As the following theorem indicates, the polynomial translation tr_1 guarantees the transfer of satisfiability from model class **M** to model class **K**.

Theorem 13 *Let $\varphi \in \mathcal{L}^{NNF}$. Then, φ is satisfiable in the class **M** if and only if $\blacksquare (\bigwedge_{\alpha \in \Gamma_{\varphi}} tr_1(\alpha)) \wedge tr_1(\varphi)$ is satisfiable in the class **K**, where Γ_{φ} is defined as follows:*

$$\begin{aligned}
\Gamma_{\varphi} = & \{now^{\geq 0}\} \cup \{now^{\geq t} \rightarrow now^{\geq t'} : t' \leq t \text{ and } now^{\geq t}, now^{\geq t'} \in SF(\varphi)\} \cup \\
& \{now^{\geq t} \leftrightarrow \Delta_{\mathfrak{m}} now^{\geq t} : now^{\geq t} \in SF(\varphi)\} \cup \\
& \{\neg now^{\geq t} \leftrightarrow \Delta_{\mathfrak{m}} \neg now^{\geq t} : now^{\geq t} \in SF(\varphi)\},
\end{aligned}$$

and $SF(\varphi)$ is the set of subformulas of φ which is inductively defined as follows:

$$\begin{aligned}
SF(p^t) &= \{p^t\}, \\
SF(\neg p^t) &= \{\neg p^t\} \cup SF(p^t), \\
SF(now^{\geq t}) &= \{now^{\geq t}\}, \\
SF(\neg now^{\geq t}) &= \{\neg now^{\geq t}\} \cup SF(now^{\geq t}), \\
SF(\Delta_h^t \alpha) &= \{\Delta_h^t \alpha\} \cup SF(\alpha), \\
SF(\neg \Delta_h^t \alpha) &= \{\neg \Delta_h^t \alpha\} \cup SF(\Delta_h^t \alpha), \\
SF(\Delta_m \alpha) &= \{\Delta_m \alpha\} \cup SF(\alpha), \\
SF(\neg \Delta_m \alpha) &= \{\neg \Delta_m \alpha\} \cup SF(\Delta_m \alpha), \\
SF(\varphi_1 \wedge \varphi_2) &= \{\varphi_1 \wedge \varphi_2\} \cup SF(\varphi_1) \cup SF(\varphi_2), \\
SF(\varphi_1 \vee \varphi_2) &= \{\varphi_1 \vee \varphi_2\} \cup SF(\varphi_1) \cup SF(\varphi_2), \\
SF(\Box_m \beta) &= \{\Box_m \beta\} \cup SF(\beta), \\
SF(\Diamond_m \beta) &= \{\Diamond_m \beta\} \cup SF(\beta).
\end{aligned}$$

As a last step, we provide a polysize reduction of \mathcal{L}^{Mod} -satisfiability to SAT, where the underlying propositional logic language \mathcal{L}^{PL} is built from the following set of atomic propositions:

$$Atm^{++} = \{q_x : q \in Atm^+ \text{ and } x \in \mathbb{N}\} \cup \{r_{x,y} : x, y \in \mathbb{N}\}.$$

Let $tr_2 : \mathcal{L}^{Mod} \times \mathbb{N} \times \mathbb{N} \longrightarrow \mathcal{L}^{PL}$ be the following translation function:

$$\begin{aligned}
tr_2(q, x, y) &= q_x, \\
tr_2(\neg \varphi, x, y) &= \neg tr_2(\varphi, x, y), \\
tr_2(\varphi_1 \wedge \varphi_2, x, y) &= tr_2(\varphi_1, x, y) \wedge tr_2(\varphi_2, x, y), \\
tr_2(\varphi_1 \vee \varphi_2, x, y) &= tr_2(\varphi_1, x, y) \vee tr_2(\varphi_2, x, y), \\
tr_2(\blacksquare \omega, x, y) &= \bigwedge_{0 \leq z \leq y} (r_{x,z} \rightarrow tr_2(\omega, z, y)), \\
tr_2(\blacklozenge \omega, x, y) &= \bigvee_{0 \leq z \leq y} (r_{x,z} \wedge tr_2(\omega, z, y)).
\end{aligned}$$

Translation tr_2 is similar to the translation of modal logic S5 into propositional logic given in [Caridroit *et al.* 2017] and, more generally, to the standard translation of modal logic into FOL in which accessibility relations are encoded by special predicates.

The size of an \mathcal{L}^{Mod} formula, $size(\varphi)$, is defined by:

$$\begin{aligned} size(p^t) &= 1, \\ size(\varphi_1 \wedge \varphi_2) &= size(\varphi_1) + size(\varphi_2) + 1, \\ size(\varphi_1 \vee \varphi_2) &= size(\varphi_1) + size(\varphi_2) + 1, \\ size(\neg\varphi) &= size(\varphi) + 1, \\ size(\blacksquare\omega) &= size(\blacklozenge\omega) \\ &= size(\omega) + 1. \end{aligned}$$

Note that the size of $tr_2(\varphi, 0, size(\varphi))$ is polynomial in the size of φ .

Theorem 14 *Let $\varphi \in \mathcal{L}^{Mod}$. Then, φ is satisfiable in the class \mathbf{K} if and only if $tr_2(\varphi, 0, size(\varphi))$ is satisfiable in propositional logic.*

Note that the size of $tr_2(\varphi, 0, size(\varphi))$ is polynomial in the size of φ . Therefore, Theorem 10 follows from Proposition 6, Theorem 13 and Theorem 14.

Detailed proof of Theorem 11

The following equivalences are valid in the class \mathbf{M} :

$$\begin{aligned} [+^t_m\alpha]\alpha' &\leftrightarrow \begin{cases} \top, & \text{if } \alpha' = \Delta_m\alpha \text{ or } (\alpha' = now^{\geq t'} \text{ and } t' \leq t) \text{ or} \\ & (\alpha' = \Delta_m now^{\geq t'} \text{ and } t' \leq t), \\ \alpha', & \text{otherwise;} \end{cases} \\ [+^t_m\alpha]\neg\varphi &\leftrightarrow \neg[+^t_m\alpha]\varphi; \\ [+^t_m\alpha](\varphi_1 \wedge \varphi_2) &\leftrightarrow [+^t_m\alpha]\varphi_1 \wedge [+^t_m\alpha]\varphi_2; \\ [+^t_m\alpha](\varphi_1 \vee \varphi_2) &\leftrightarrow [+^t_m\alpha]\varphi_1 \vee [+^t_m\alpha]\varphi_2; \\ [+^t_m\alpha]\Box_m\alpha' &\leftrightarrow \Box_m((\alpha \wedge \bigwedge_{t' \leq t} now^{\geq t'}) \rightarrow \alpha'); \\ [+^t_m\alpha]\Diamond_m\alpha' &\leftrightarrow \Diamond_m((\alpha \wedge \bigwedge_{t' \leq t} now^{\geq t'}) \wedge \alpha'). \end{aligned}$$

Thanks to these equivalences we can define the following reduction *red* trans-

forming every \mathcal{L}^+ formula φ into an equivalent \mathcal{L} formula $red(\varphi)$:

$$\begin{aligned}
red(p^t) &= p^t, \\
red(\Delta_h^t \alpha) &= \Delta_h^t \alpha, \\
red(\Delta_m \alpha) &= \Delta_m \alpha, \\
red(\neg \varphi) &= \neg red(\varphi), \\
red(\varphi_1 \wedge \varphi_2) &= red(\varphi_1) \wedge red(\varphi_2), \\
red(\varphi_1 \vee \varphi_2) &= red(\varphi_1) \vee red(\varphi_2), \\
red(\Box_m \varphi) &= \Box_m red(\varphi), \\
red(\Diamond_m \varphi) &= \Diamond_m red(\varphi), \\
red([+^t_m \alpha] \alpha') &= \begin{cases} \top, & \text{if } \alpha' = \Delta_m \alpha \text{ or } (\alpha' = now^{\geq t'} \text{ and } t' \leq t) \text{ or} \\ & (\alpha' = \Delta_m now^{\geq t'} \text{ and } t' \leq t), \\ red(\alpha'), & \text{otherwise;} \end{cases} \\
red([+^t_m \alpha] \neg \varphi) &= red(\neg [+^t_m \alpha] \varphi), \\
red([+^t_m \alpha] (\varphi_1 \wedge \varphi_2)) &= red([+^t_m \alpha] \varphi_1 \wedge [+^t_m \alpha] \varphi_2), \\
red([+^t_m \alpha] (\varphi_1 \vee \varphi_2)) &= red([+^t_m \alpha] \varphi_1 \vee [+^t_m \alpha] \varphi_2), \\
red([+^t_m \alpha] \Box_m \alpha') &= red\left(\Box_m \left((\alpha \wedge \bigwedge_{t' \leq t} now^{\geq t'}) \rightarrow \alpha'\right)\right); \\
red([+^t_m \alpha] \Diamond_m \alpha') &= red\left(\Diamond_m \left((\alpha \wedge \bigwedge_{t' \leq t} now^{\geq t'}) \wedge \alpha'\right)\right).
\end{aligned}$$

Proposition 7 *Let $\varphi \in \mathcal{L}^+$. Then, $\varphi \leftrightarrow red(\varphi)$ is valid in the class \mathbf{M} , and $red(\varphi) \in \mathcal{L}$.*

Theorem 11 is a consequence of Theorem 10, Proposition 7 and the fact that the size of $red(\varphi)$ is polynomial in the size of φ .

The separation constraint

For every $p, p' \in GRID : p \neq p'$, $t \in TIME$, and $S \subseteq GRID : p, p' \notin S$:

$$\begin{aligned}
linked_{p,p',S}^t &\stackrel{\text{def}}{=} \left(\bigvee_{1 \leq i \leq |S|} path_{(p,p'),p',i}^t \right) \wedge \bigwedge_{p'' \in S \setminus NEIG(p)} \neg path_{(p,p'),p'',1}^t \\
&\wedge \bigwedge_{\substack{2 \leq i \leq |S| \\ p'' \in S}} \left(path_{(p,p'),p'',i}^t \rightarrow \bigvee_{p''' \in NEIG(p'')} path_{(p,p'),p''',i-1}^t \right)
\end{aligned}$$

that reads: “there is a path in S from position p to position p' iff: (1) p' is reachable in at most $|S|$ steps, (2) no position which is not a neighbor of p in S is reachable

from p in 1 step, (3) any position p'' in the path is reachable from p within i steps only if there is at least a neighbor p''' of p'' in S which is reachable in $i - 1$ steps.”

Finally, for $p \in GRID$, and $x \in CARDS$:

$$\begin{aligned}
 OPOS^t(S) &\stackrel{\text{def}}{=} \bigwedge_{p \in S} \neg \text{emp}_p^t \wedge \bigwedge_{p \in GRID \setminus S} \text{emp}_p^t \\
 \text{legMov}_{x,p}^t &\stackrel{\text{def}}{=} \bigvee_{\substack{p'' \in GRID \\ p'' \neq p}} \left(\text{pos}_{x,p''}^t \wedge \text{emp}_p^t \wedge \right. \\
 &\quad \left. \bigwedge_{p' \in NEIG(p'')} \bigvee_{S \in 2^{GRID}} \left(OPOS^t(S) \wedge \text{linked}_{p,p',S \setminus \{p''\}}^t \right) \right)
 \end{aligned}$$

emp_p^t reads “the position p is empty”; $OPOS^t$ is the set of positions occupied by a card at time t ; $\text{legMov}_{x,p}^t$ reads “the move of card x towards position p at time t is authorized by the rules of the game iff p is currently an empty position and there is a sequence of adjacent positions between p and any other occupied positions (except the initial position p'' of card x) through the set of currently occupied positions (excluding p'').

Appendix B

Implementation of Yōkai

This appendix is devoted to explaining the implementation of the Yōkai board game. In the system architecture presented in Figure 6, it is possible to identify three main modules: the GUI (front-end) and the belief revision and planning modules (back-end). The GUI allows the players to perform their actions at each turn, while the back-end modules update the belief base and generate the actions corresponding to agent m . During the game, players can observe in the GUI the actions performed by the other player.

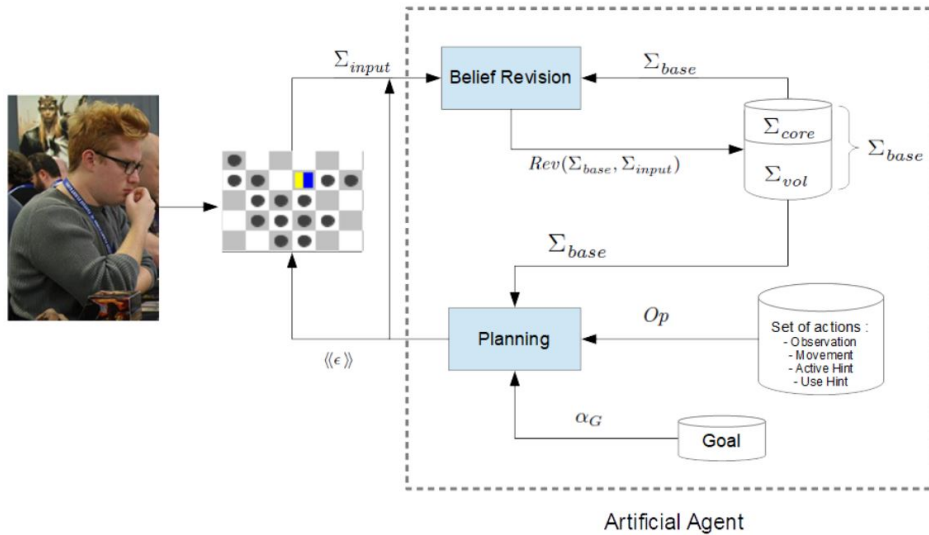


Figure 6: System architecture

At each time point, the GUI module generates the text file `/sdata/01_ini_set.txt` (Listing 1). This file contains information related to the positions of the cards, neighbors' cards, occupied positions, empty positions on the game perimeter, active hints, and hints colors. The function that generates this file has been implemented in the GUI program inside the `Translator.java` class. In fact, by generating this information and performing some validations, the GUI plays the role of the *third agent* that we described in Section 5.7.

```

1 $now = 0
2 $length = ($now + 1)
3 $I = [pos_1_p_12_12, pos_2_p_12_13, pos_3_p_12_14, pos_4_p_12_15,
      pos_5_p_13_12, pos_6_p_13_13, pos_7_p_13_14, pos_8_p_13_15, pos_9_p_14_12,
      pos_10_p_14_13, pos_11_p_14_14, pos_12_p_14_15, pos_13_p_15_12,
      pos_14_p_15_13, pos_15_p_15_14, pos_16_p_15_15]

```

```

4 $POSINI(1) = [p_12_12]
5 $POSINI(2) = [p_12_13]
6 $POSINI(3) = [p_12_14]
7 $POSINI(4) = [p_12_15]
8 $POSINI(5) = [p_13_12]
9 $POSINI(6) = [p_13_13]
10 $POSINI(7) = [p_13_14]
11 $POSINI(8) = [p_13_15]
12 $POSINI(9) = [p_14_12]
13 $POSINI(10) = [p_14_13]
14 $POSINI(11) = [p_14_14]
15 $POSINI(12) = [p_14_15]
16 $POSINI(13) = [p_15_12]
17 $POSINI(14) = [p_15_13]
18 $POSINI(15) = [p_15_14]
19 $POSINI(16) = [p_15_15]
20 $NEIG(p_12_12) = [p_11_12, p_12_11]
21 $NEIG(p_12_13) = [p_11_13]
22 $NEIG(p_12_14) = [p_11_14]
23 $NEIG(p_12_15) = [p_11_15, p_12_16]
24 $NEIG(p_13_12) = [p_13_11]
25 $NEIG(p_13_13) = []
26 $NEIG(p_13_14) = []
27 $NEIG(p_13_15) = [p_13_16]
28 $NEIG(p_14_12) = [p_14_11]
29 $NEIG(p_14_13) = []
30 $NEIG(p_14_14) = []
31 $NEIG(p_14_15) = [p_14_16]
32 $NEIG(p_15_12) = [p_16_12, p_15_11]
33 $NEIG(p_15_13) = [p_16_13]
34 $NEIG(p_15_14) = [p_16_14]
35 $NEIG(p_15_15) = [p_16_15, p_15_16]
36 $NEIGBOR(p_12_12) = [p_11_12, p_13_12, p_12_13, p_12_11]
37 $NEIGBOR(p_12_13) = [p_11_13, p_13_13, p_12_14, p_12_12]
38 $NEIGBOR(p_12_14) = [p_11_14, p_13_14, p_12_15, p_12_13]
39 $NEIGBOR(p_12_15) = [p_11_15, p_13_15, p_12_16, p_12_14]
40 $NEIGBOR(p_13_12) = [p_12_12, p_14_12, p_13_13, p_13_11]
41 $NEIGBOR(p_13_13) = [p_12_13, p_14_13, p_13_14, p_13_12]
42 $NEIGBOR(p_13_14) = [p_12_14, p_14_14, p_13_15, p_13_13]
43 $NEIGBOR(p_13_15) = [p_12_15, p_14_15, p_13_16, p_13_14]
44 $NEIGBOR(p_14_12) = [p_13_12, p_15_12, p_14_13, p_14_11]
45 $NEIGBOR(p_14_13) = [p_13_13, p_15_13, p_14_14, p_14_12]
46 $NEIGBOR(p_14_14) = [p_13_14, p_15_14, p_14_15, p_14_13]
47 $NEIGBOR(p_14_15) = [p_13_15, p_15_15, p_14_16, p_14_14]
48 $NEIGBOR(p_15_12) = [p_14_12, p_16_12, p_15_13, p_15_11]
49 $NEIGBOR(p_15_13) = [p_14_13, p_16_13, p_15_14, p_15_12]
50 $NEIGBOR(p_15_14) = [p_14_14, p_16_14, p_15_15, p_15_13]
51 $NEIGBOR(p_15_15) = [p_14_15, p_16_15, p_15_16, p_15_14]
52 $HINTS = [1, 2, 3, 4, 5, 6, 7]
53 $AVHINTS = []
54 $ACHINTS = []
55 $HINT_COLOR(1) = [B]
56 $HINT_COLORSS(1,1) = [B]
57 $HINT_COLOR(2) = [B, G]
58 $HINT_COLORSS(2,1) = [B]
59 $HINT_COLORSS(2,2) = [G]
60 $HINT_COLOR(3) = [R, B, Y]
61 $HINT_COLORSS(3,1) = [R]
62 $HINT_COLORSS(3,2) = [B]
63 $HINT_COLORSS(3,3) = [Y]
64 $HINT_COLOR(4) = [R]
65 $HINT_COLORSS(4,1) = [R]

```

```

66 $HINT_COLOR(5) = [Y, G]
67 $HINT_COLORSS(5,1) = [Y]
68 $HINT_COLORSS(5,2) = [G]
69 $HINT_COLOR(6) = [G, R, Y]
70 $HINT_COLORSS(6,1) = [G]
71 $HINT_COLORSS(6,2) = [R]
72 $HINT_COLORSS(6,3) = [Y]
73 $HINT_COLOR(7) = [G, B]
74 $HINT_COLORSS(7,1) = [G]
75 $HINT_COLORSS(7,2) = [B]
76 $COLORS = [R,G,B,Y]
77 $OPOS($now) = [p_12_12, p_12_13, p_12_14, p_12_15, p_13_12, p_13_13, p_13_14,
    p_13_15, p_14_12, p_14_13, p_14_14, p_14_15, p_15_12, p_15_13, p_15_14,
    p_15_15]
78 $PERIM = [p_11_12, p_12_11, p_11_13, p_11_14, p_11_15, p_12_16, p_13_11,
    p_13_16, p_14_11, p_14_16, p_16_12, p_15_11, p_16_13, p_16_14, p_16_15,
    p_15_16]
79 $GRID = [p_12_12, p_13_12, p_14_12, p_15_12, p_15_13, p_15_14, p_15_15,
    p_14_15, p_13_15, p_12_15, p_12_14, p_12_13]
80 $CARDS = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

```

Listing 1: Sets file generated by the GUI module

Formulas representing Σ_c are saved in the file `/sdata/00_axioms.touist` as it is shown in Listing 2.

```

1 ;; (SAT-EFA1) Etat initial et but
2 bigand $i in $I: $i($now) end
3
4 ;; ICP1
5 bigand $t in [$now..$length]:
6   bigand $x in $CARDS :
7     bigor $p in $GRID :
8       pos($x,$p,$t)
9     end
10  end
11 end
12 ;;END_ICP1
13
14 ;; ICP2
15 bigand $t in [$now..$length]:
16   bigand $x,$p1,$p2 in $CARDS,$CARDS,$GRID when $p1 != $p2:
17     not (pos($x,$p1,$t) and pos($x,$p2,$t))
18   end
19 end
20 ;;END_ICP2
21
22 ;; ICP3
23 bigand $t in [$now..$length]:
24   bigand $x1,$x2,$p in $CARDS,$CARDS,$GRID when $x1 != $x2:
25     not (pos($x1,$p,$t) and pos($x2,$p,$t))
26   end
27 end
28 ;;END_ICP3
29 ;; ICC4
30 bigand $t in [$now..$length]:
31   bigand $x in $CARDS :
32     bigor $c in $COLORS :
33       col($x,$c,$t)
34   end
35 end

```

```

36 end
37 ;;END_ICC4
38
39 ;;ICC5
40 bigand $t in [$now..$length]:
41   bigand $x,$c1,$c2 in $CARDS,$COLORS,$COLORS when $c1 != $c2:
42     not (col($x,$c1,$t) and col($x,$c2,$t))
43   end
44 end
45 ;;END_ICC5
46
47 ;;SSA11
48 bigand $t in [$now+1..$length]:
49   bigand $xa, $ap, $movact in $CARDS, $POSINI($xa), $NEIG($ap) :
50     (pos($xa,$movact,$t) and not pos($xa,$movact,$t-1))
51     => ((bigand $x, $p in $CARDS diff [$xa], $POSINI($x) : pos($x,$p,$t)
52         <=>
53           pos($x,$p,$t-1) end) and
54         (bigand $x, $c in $CARDS, $COLORS : (col($x,$c,$t) <=>
55           col($x,$c,$t-1)) and (not tm_col($x,$c,$t) <=>
56           not tm_col($x,$c,$t-1)) end) and
57         (bigand $x, $h in $CARDS, $AVHINTS : mark($x,$h,$t) <=>
58           mark($x,$h,$t-1) end))
59   end
60 ;;END_SSA11
61
62 ;;SSA11
63 bigand $t in [$now+1..$length]:
64   bigand $xa, $coloract in $CARDS, $COLORS :
65     (col($xa,$coloract,$t) and not tm_col($xa,$coloract,$t-1))
66     => ( (bigand $x, $p in $CARDS, $POSINI($x) : pos($x,$p,$t) <=>
67         pos($x,$p,$t-1) end) and
68         (bigand $x, $c in $CARDS diff [$xa], $COLORS : (col($x,$c,$t) <=>
69           col($x,$c,$t-1)) or (not tm_col($x,$c,$t) <=>
70           not tm_col($x,$c,$t-1)) end) and
71         (bigand $x, $h in $CARDS, $AVHINTS : mark($x,$h,$t) <=>
72           mark($x,$h,$t-1) end))
73   end
74 end
75 ;;END_SSA11
76
77 ;;SSA12
78 bigand $t in [$now+1..$length]:
79   bigand $hintact in $HINTS :
80     (act($hintact,$t) and not act($hintact,$t-1))
81     => ( (bigand $x, $p in $CARDS, $GRID : pos($x,$p,$t) <=>
82         pos($x,$p,$t-1) end) and
83         (bigand $x, $c in $CARDS, $COLORS : (col($x,$c,$t) <=>
84           col($x,$c,$t-1)) and (not tm_col($x,$c,$t) <=>
85           not tm_col($x,$c,$t-1)) end) and
86         (bigand $h in $HINTS diff $ACHINTS : act($h,$t) <=> act($h,$t-1) end
87       )
88   )
89 end
90 ;;END_SSA12
91
92 ;;SSA13
93 bigand $t in [$now+1..$length]:
94   bigand $xa, $hintact in $CARDS, $HINTS :
95     (mark($xa,$hintact,$t) and not mark($xa,$hintact,$t-1))

```

```

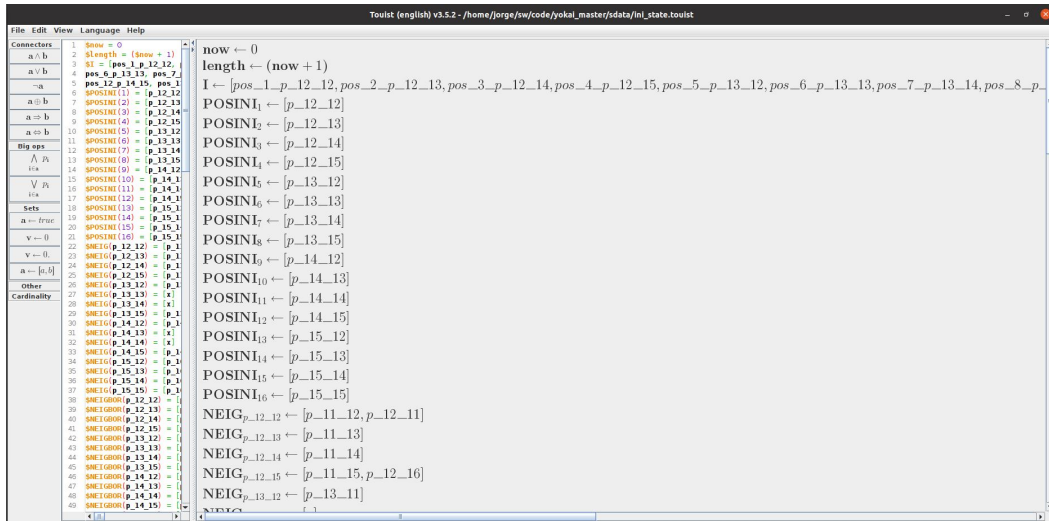
96 => ((bigand $x, $p in $CARDS, $GRID : pos($x,$p,$t) <=>
97     pos($x,$p,$t-1) end) and
98     (bigand $x, $h in $CARDS diff [$xa], $AVHINTS : mark($x,$h,$t) <=>
99     mark($x,$h,$t-1) end))
100 end
101 end
102 ;;END_SSA13

```

Listing 2: Formulas representing core beliefs

Formulas in Σ_c uses the sets of information provided by the file:/sdata/01_ini_set.txt. Thus, the next step is to join the information of both files. The sets of information provided by the file:/sdata/01_ini_set.txt are inserted dynamically at the top of the axioms file: /sdata/00_axioms.touist at each time point.

We use TouIST to verify the syntax of the logical formulas. Next, we can observe the validation of the sets and formulas in the TouIST GUI.



The screenshot shows the TouIST GUI with a list of formulas and their validation status. The formulas are organized into several categories:

- Connectors:**
 - 1 show = 0
 - 2 !!(now) = (show + 1)
 - 3 !t = pos_1_p_12_12
 - 4 pos_6_p_12_13_pos_7
 - 5 pos_12_p_14_15_pos_1
- Logic:**
 - 6 sposind(1) = [p_12_12]
 - 7 sposind(2) = [p_12_13]
 - 8 sposind(3) = [p_12_14]
 - 9 sposind(4) = [p_12_15]
 - 10 sposind(5) = [p_13_12]
 - 11 sposind(6) = [p_13_13]
 - 12 sposind(7) = [p_13_14]
 - 13 sposind(8) = [p_13_15]
 - 14 sposind(9) = [p_14_12]
 - 15 sposind(10) = [p_14_13]
 - 16 sposind(11) = [p_14_14]
 - 17 sposind(12) = [p_14_15]
 - 18 sposind(13) = [p_15_12]
 - 19 sposind(14) = [p_15_13]
 - 20 sposind(15) = [p_15_14]
 - 21 sposind(16) = [p_15_15]
- Big ops:**
 - 22 snet(p_12_12) = [p_1]
 - 23 snet(p_12_13) = [p_1]
 - 24 snet(p_12_14) = [p_1]
 - 25 snet(p_12_15) = [p_1]
 - 26 snet(p_13_12) = [p_1]
 - 27 snet(p_13_13) = [p_1]
 - 28 snet(p_13_14) = [p_1]
 - 29 snet(p_13_15) = [p_1]
 - 30 snet(p_14_12) = [p_1]
 - 31 snet(p_14_13) = [p_1]
 - 32 snet(p_14_14) = [p_1]
 - 33 snet(p_14_15) = [p_1]
 - 34 snet(p_15_12) = [p_1]
 - 35 snet(p_15_13) = [p_1]
 - 36 snet(p_15_14) = [p_1]
 - 37 snet(p_15_15) = [p_1]
 - 38 snetgrid(p_12_12) = [p_1]
 - 39 snetgrid(p_12_13) = [p_1]
 - 40 snetgrid(p_12_14) = [p_1]
 - 41 snetgrid(p_12_15) = [p_1]
 - 42 snetgrid(p_13_12) = [p_1]
 - 43 snetgrid(p_13_13) = [p_1]
 - 44 snetgrid(p_13_14) = [p_1]
 - 45 snetgrid(p_13_15) = [p_1]
 - 46 snetgrid(p_14_12) = [p_1]
 - 47 snetgrid(p_14_13) = [p_1]
 - 48 snetgrid(p_14_14) = [p_1]
 - 49 snetgrid(p_14_15) = [p_1]
- Other:**
 - 50 snet(p_12_12) = [p_1]
 - 51 snet(p_12_13) = [p_1]
 - 52 snet(p_12_14) = [p_1]
 - 53 snet(p_12_15) = [p_1]
 - 54 snet(p_13_12) = [p_1]
 - 55 snet(p_13_13) = [p_1]
 - 56 snet(p_13_14) = [p_1]
 - 57 snet(p_13_15) = [p_1]
 - 58 snet(p_14_12) = [p_1]
 - 59 snet(p_14_13) = [p_1]
 - 60 snet(p_14_14) = [p_1]
 - 61 snet(p_14_15) = [p_1]
 - 62 snet(p_15_12) = [p_1]
 - 63 snet(p_15_13) = [p_1]
 - 64 snet(p_15_14) = [p_1]
 - 65 snet(p_15_15) = [p_1]
- Cardinality:**
 - 66 snet(p_12_12) = [p_1]
 - 67 snet(p_12_13) = [p_1]
 - 68 snet(p_12_14) = [p_1]
 - 69 snet(p_12_15) = [p_1]
 - 70 snet(p_13_12) = [p_1]
 - 71 snet(p_13_13) = [p_1]
 - 72 snet(p_13_14) = [p_1]
 - 73 snet(p_13_15) = [p_1]
 - 74 snet(p_14_12) = [p_1]
 - 75 snet(p_14_13) = [p_1]
 - 76 snet(p_14_14) = [p_1]
 - 77 snet(p_14_15) = [p_1]
 - 78 snet(p_15_12) = [p_1]
 - 79 snet(p_15_13) = [p_1]
 - 80 snet(p_15_14) = [p_1]
 - 81 snet(p_15_15) = [p_1]

The right side of the GUI shows the following formulas and their validation status:

```

now ← 0
length ← (now + 1)
I ← [pos_1_p_12_12, pos_2_p_12_13, pos_3_p_12_14, pos_4_p_12_15, pos_5_p_13_12, pos_6_p_13_13, pos_7_p_13_14, pos_8_p_13_15, pos_9_p_14_12, pos_10_p_14_13, pos_11_p_14_14, pos_12_p_14_15, pos_13_p_15_12, pos_14_p_15_13, pos_15_p_15_14, pos_16_p_15_15]
POSINI_1 ← [p_12_12]
POSINI_2 ← [p_12_13]
POSINI_3 ← [p_12_14]
POSINI_4 ← [p_12_15]
POSINI_5 ← [p_13_12]
POSINI_6 ← [p_13_13]
POSINI_7 ← [p_13_14]
POSINI_8 ← [p_13_15]
POSINI_9 ← [p_14_12]
POSINI_10 ← [p_14_13]
POSINI_11 ← [p_14_14]
POSINI_12 ← [p_14_15]
POSINI_13 ← [p_15_12]
POSINI_14 ← [p_15_13]
POSINI_15 ← [p_15_14]
POSINI_16 ← [p_15_15]
NEIG_p_12_12 ← [p_11_12, p_12_11]
NEIG_p_12_13 ← [p_11_13]
NEIG_p_12_14 ← [p_11_14]
NEIG_p_12_15 ← [p_11_15, p_12_16]
NEIG_p_13_12 ← [p_13_11]

```

Figure 7: Checking syntax for formulas in Σ_c using TouIST GUI

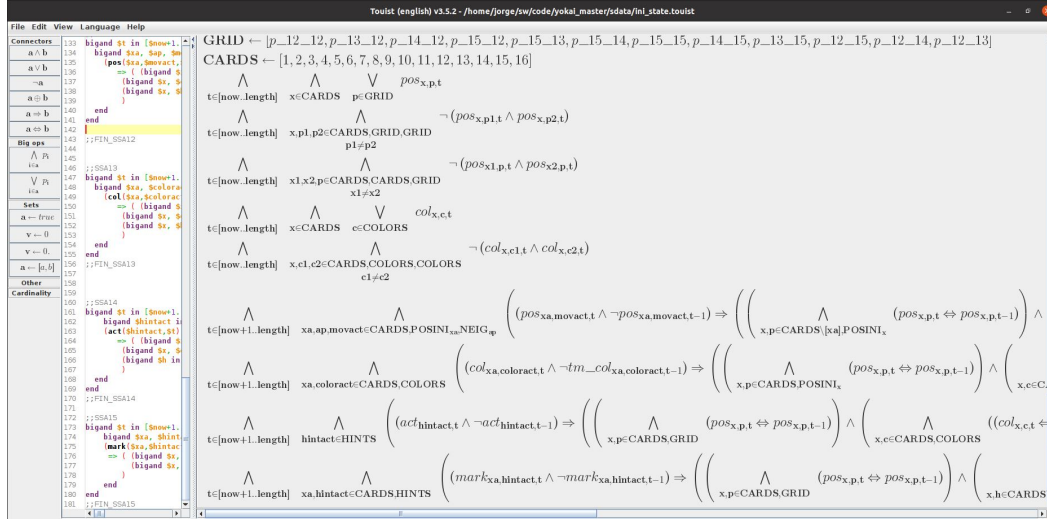


Figure 8: Checking syntax for formulas in Σ_c using TouIST GUI

The initial mutable beliefs are saved in the file `/sdata/00_axioms.touist` as it is shown in Listing 3.

```

1 now_t0
2 not tm_col_1_G_t0
3 not tm_col_2_B_t0
4 not tm_col_3_R_t0
5 not tm_col_4_R_t0
6 not tm_col_5_G_t0
7 not tm_col_6_R_t0
8 not tm_col_7_B_t0
9 not tm_col_8_Y_t0
10 not tm_col_9_Y_t0
11 not tm_col_10_Y_t0
12 not tm_col_11_Y_t0
13 not tm_col_12_G_t0
14 not tm_col_13_B_t0
15 not tm_col_14_G_t0
16 not tm_col_15_B_t0
17 not tm_col_16_R_t0
18 not th_col_1_B_t0 and not th_col_1_G_t0 and not th_col_1_R_t0 and not
   th_col_1_Y_t0
19 not th_col_2_B_t0 and not th_col_2_G_t0 and not th_col_2_R_t0 and not
   th_col_2_Y_t0
20 not th_col_3_B_t0 and not th_col_3_G_t0 and not th_col_3_R_t0 and not
   th_col_3_Y_t0
21 not th_col_4_B_t0 and not th_col_4_G_t0 and not .
22 .
23 .
24 .
25 not th_col_15_B_t0 and not th_col_15_G_t0 and not th_col_15_R_t0 and not
   th_col_15_Y_t0
26 not th_col_16_B_t0 and not th_col_16_G_t0 and not th_col_16_R_t0 and not
   th_col_16_Y_t0
27 not act_1_t0
28 not act_2_t0
29 not act_3_t0
30 not act_4_t0
31 not act_5_t0

```

```

32 not act_6_t0
33 not act_7_t0
34 not mark_1_1_t0
35 not mark_1_2_t0
36 .
37 .
38 .
39 not mark_16_6_t0
40 not mark_16_7_t0
41 pos_1_p_12_12_t0
42 pos_2_p_12_13_t0
43 pos_3_p_12_14_t0
44 pos_4_p_12_15_t0
45 pos_5_p_13_12_t0
46 pos_6_p_13_13_t0
47 pos_7_p_13_14_t0
48 pos_8_p_13_15_t0
49 pos_9_p_14_12_t0
50 pos_10_p_14_13_t0
51 pos_11_p_14_14_t0
52 pos_12_p_14_15_t0
53 pos_13_p_15_12_t0
54 pos_14_p_15_13_t0
55 pos_15_p_15_14_t0
56 pos_16_p_15_15_t0

```

Listing 3: Mutable beliefs

Thereafter, the system joins the previous files containing Σ_c and Σ_m . This union will represent agent m 's belief base Σ we illustrate in the architecture sketched in Figure 6.

At this point, it is worth mentioning that when we used TouIST to obtain the Abstract Syntax Tree (AST)⁶ from the set of formulas in `00_axioms.touist` and we saved the AST in a text file, the generated file had approximately 6 megabytes size on average. This size is a significant amount of data for a text file. When we traced the source of this problem by evaluating the AST, we found that the reason was that some formulas contained in Σ_c generated too many combinations.

An example of this situation is formula ICC6 detailed in section 5.5.1. According to this there should be a maximum of 4 cards of the same color. The formula tries to implement this constraint by generating all possible combinations of 4 cards among 16 cards. The resulting combination produces 1,820 possibilities by color, so when we multiply by four existing colors, the resulting combinations increase to 7,280 possibilities only for this formula.

Similarly, formula ICP1 states that each card has a position, and that position must be unique in the grid. If we have a board with 32 x 32 positions, then for each card, we have 1024 possible positions multiplied by 16 cards. Thus, the possible positions for the total cards are 16,384.

We consider it redundant to implement formula ICC6 because the GUI will never generate more than four cards of the same color. Similarly, in the case of formula ICP1, the GUI will assign each card a unique position in the grid at each time

⁶<https://www.irit.fr/TouIST/wp-content/uploads/2020/04/reference-manual.html>

point. We propose implementing these formulas at the GUI level and not in the logical language to avoid overflow and reduce the computation time.

Furthermore, some functions also represent a big challenge to implement using purely logical formulas. An example of this situation is the function $legMov_{x,p}^t$ used to validate the separation constraint 5.5.2.

In order to simplify the implementation and reduce the programming time, we included this function in the Yōkai GUI :

```
legalmove = game.move(out, (Position)computerActionx.get(2));
```

The previous function reads the new position and verifies if this has a common side with another card in the grid using the following java code:

```
1 public int adjacent(Card c, Position pos) {
2   int numberAdjacent = 0;
3   if (pos.getX() - 1 >= 0 && cardList[pos.getX() - 1][pos.getY()] != null
4       && !(cardList[pos.getX() - 1][pos.getY()].equals(c)))
5     numberAdjacent++;
6   if (pos.getX() + 1 < LENGTH && cardList[pos.getX() + 1][pos.getY()] != null
7       && !(cardList[pos.getX() + 1][pos.getY()].equals(c)))
8     numberAdjacent++;
9   if (pos.getY() + 1 >= 0 && cardList[pos.getX()][pos.getY() + 1] != null
10      && !(cardList[pos.getX()][pos.getY() + 1].equals(c)))
11     numberAdjacent++;
12  if (pos.getY() - 1 < LENGTH && cardList[pos.getX()][pos.getY() - 1] != null
13      && !(cardList[pos.getX()][pos.getY() - 1].equals(c)))
14     numberAdjacent++;
15  return numberAdjacent;
16 }
```

Listing 4: Legal move function in the GUI

The GUI executes the *legalmove* function passing the target position as parameter. The resulting value is assigned to the boolean variable `legalmove`. For example in the scenario illustrated in Figure 9, the machine intends to move card 3, from the current position `pos_3_p_12_14` to the final position `pos_3_p_11_12`.

In this case the final position is linked with at least one card in the grid (card 1). Consequently the `legalmove` function is equal to *True*. Based on that, the GUI will repaint the board to show the new cards' positions.

Otherwise, the system will rerun the planning module until it can find a valid target position. Moreover, at each iteration the system will excluded illegal target positions generated by previous plans from the set of available actions to prevent an infinite loop.

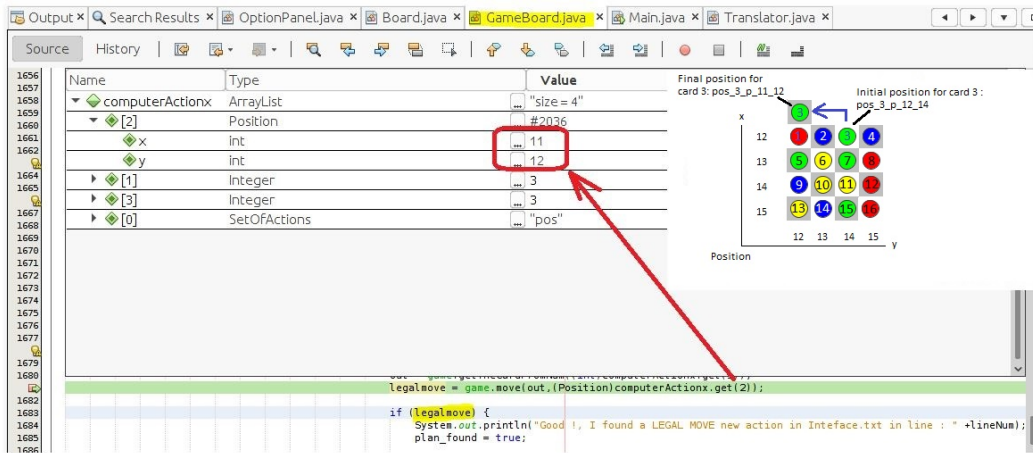


Figure 9: Legal move function

When the game starts, the cards are placed forming a square in the middle of the board. The cards are randomly colored, as shown in Figure 10. Concerning the initial positions, card 1 is always placed on the upper left corner of the square at position p_{12_12} (axis $x = 12$ and axis $y = 12$). Similarly, card 16 will be located on the lower right corner of the square at position p_{15_15} (axis $x = 15$ and axis $y = 15$).

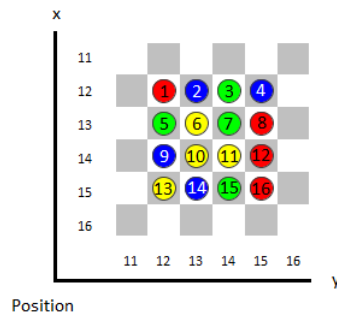


Figure 10: Cards distribution at the beginning of the game

In Figure 11 we can see the section of Σ_m that contains the beliefs resulting from the actions performed during agent m 's round 1.

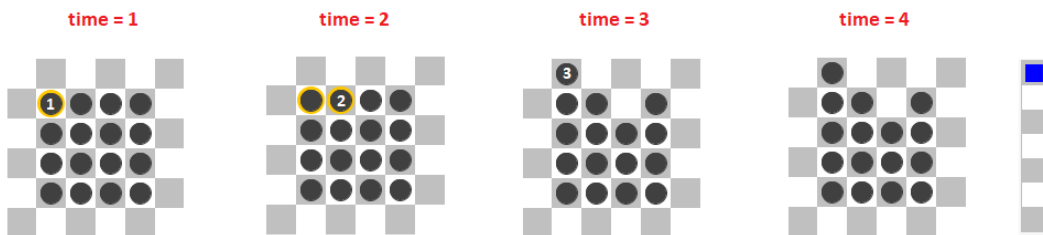


Figure 11: Agent m 's first round

At slot time 1, agent m 's observes card 1:

```
1 col_1_R_t1 and tm_col_1_R_t1 and (th_tm_col_1_G_t1 or th_tm_col_1_Y_t1 or
  th_tm_col_1_B_t1 or th_tm_col_1_R_t1)
```

Listing 5: Σ_m entries at time point 1

At slot time 2, agent m 's observes card 2:

```
1 col_1_R_t2 and tm_col_1_R_t2 and (th_tm_col_1_G_t2 or th_tm_col_1_Y_t2 or
  th_tm_col_1_B_t2 or th_tm_col_1_R_t2)
2 col_2_B_t2 and tm_col_2_B_t2 and (th_tm_col_2_G_t2 or th_tm_col_2_Y_t2 or
  th_tm_col_2_B_t2 or th_tm_col_2_R_t2)
```

Listing 6: Σ_m entries at time point 2

At slot time 3, agent m 's moves card 3 from `pos_12_14` to `pos_11_12`:

```
1 col_1_R_t3 and tm_col_1_R_t3 and (th_tm_col_1_G_t3 or th_tm_col_1_Y_t3 or
  th_tm_col_1_B_t3 or th_tm_col_1_R_t3)
2 col_2_B_t3 and tm_col_2_B_t3 and (th_tm_col_2_G_t3 or th_tm_col_2_Y_t3 or
  th_tm_col_2_B_t3 or th_tm_col_2_R_t3)
3 pos_3_p_11_12_t3 and tm_pos_3_p_11_12_t3 and th_pos_3_p_11_12_t3
```

Listing 7: Σ_m entries at time point 3

At time 4, agent m 's activates a hint:

```
1 col_1_R_t4 and tm_col_1_R_t4 and (th_tm_col_1_G_t4 or th_tm_col_1_Y_t4 or
  th_tm_col_1_B_t4 or th_tm_col_1_R_t4)
2 col_2_B_t4 and tm_col_2_B_t4 and (th_tm_col_2_G_t4 or th_tm_col_2_Y_t4 or
  th_tm_col_2_B_t4 or th_tm_col_2_R_t4)
3 pos_3_p_11_12_t4 and tm_pos_3_p_11_12_t4 and th_pos_3_p_11_12_t4
4 act_1_t4 and tm_act_1_t4 and th_act_1_t4
```

Listing 8: Σ_m entries at time point 4

Once agent m has finished its round, it is time to play for agent h . In Figure 12, we illustrate an example of a sequence of actions performed during agent h 's first round.

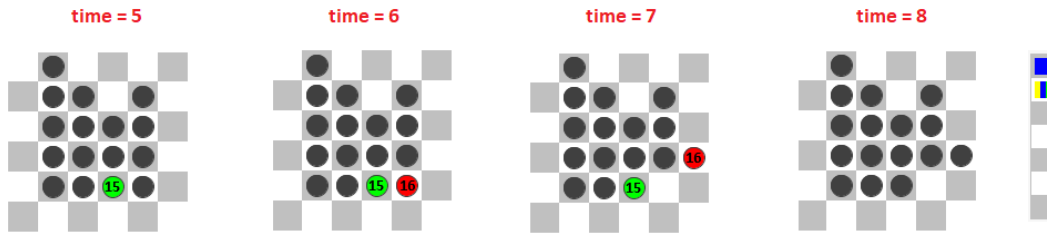


Figure 12: Agent h 's first round

At slot time 5, agent h 's observes card 15:

```
1 col_1_R_t5 and tm_col_1_R_t5 and (th_tm_col_1_G_t5 or th_tm_col_1_Y_t5 or
  th_tm_col_1_B_t5 or th_tm_col_1_R_t5)
2 col_2_B_t5 and tm_col_2_B_t5 and (th_tm_col_2_G_t5 or th_tm_col_2_Y_t5 or
  th_tm_col_2_B_t5 or th_tm_col_2_R_t5)
3 pos_3_p_11_12_t5 and tm_pos_3_p_11_12_t5 and th_pos_3_p_11_12_t5
4 act_1_t5 and tm_act_1_t5 and th_act_1_t5
```

```
5 (th_col_15_R_t5 or th_col_15_G_t5 or th_col_15_Y_t5 or th_col_15_B_t5)
```

Listing 9: Σ_m entries at time point 5

At slot time 6, agent h 's observes card 16:

```
1 col_1_R_t6 and tm_col_1_R_t6 and (th_tm_col_1_G_t6 or th_tm_col_1_Y_t6 or
  th_tm_col_1_B_t6 or th_tm_col_1_R_t6)
2 col_2_B_t6 and tm_col_2_B_t6 and (th_tm_col_2_G_t6 or th_tm_col_2_Y_t6 or
  th_tm_col_2_B_t6 or th_tm_col_2_R_t6)
3 pos_3_p_11_12_t6 and tm_pos_3_p_11_12_t6 and th_pos_3_p_11_12_t6
4 act_1_t6 and tm_act_1_t6 and th_act_1_t6
5 (th_col_15_R_t6 or th_col_15_G_t6 or th_col_15_Y_t6 or th_col_15_B_t6)
6 (th_col_16_R_t6 or th_col_16_G_t6 or th_col_16_Y_t6 or th_col_16_B_t6)
```

Listing 10: Σ_m entries at time point 6

At slot time 7, agent h moves card 16 from `pos_15_15` to `pos_14_16`:

```
1 col_1_R_t7 and tm_col_1_R_t7 and (th_tm_col_1_G_t7 or th_tm_col_1_Y_t7 or
  th_tm_col_1_B_t7 or th_tm_col_1_R_t7)
2 col_2_B_t7 and tm_col_2_B_t7 and (th_tm_col_2_G_t7 or th_tm_col_2_Y_t7 or
  th_tm_col_2_B_t7 or th_tm_col_2_R_t7)
3 pos_3_p_11_12_t7 and tm_pos_3_p_11_12_t7 and th_pos_3_p_11_12_t7
4 act_1_t7 and tm_act_1_t7 and th_act_1_t7
5 (th_col_15_R_t7 or th_col_15_G_t7 or th_col_15_Y_t7 or th_col_15_B_t7)
6 (th_col_16_R_t7 or th_col_16_G_t7 or th_col_16_Y_t7 or th_col_16_B_t7)
7 pos_16_p_14_16_t7 and tm_pos_16_p_14_16_t7 and th_pos_16_p_14_16_t7
```

Listing 11: Σ_m entries at time point 7

At slot time 8, agent h activates a hint:

```
1 col_1_R_t8 and tm_col_1_R_t8 and (th_tm_col_1_G_t8 or th_tm_col_1_Y_t8 or
  th_tm_col_1_B_t8 or th_tm_col_1_R_t8)
2 col_2_B_t8 and tm_col_2_B_t8 and (th_tm_col_2_G_t8 or th_tm_col_2_Y_t8 or
  th_tm_col_2_B_t8 or th_tm_col_2_R_t8)
3 pos_3_p_11_12_t8 and tm_pos_3_p_11_12_t8 and th_pos_3_p_11_12_t8
4 act_1_t8 and tm_act_1_t8 and th_act_1_t8
5 (th_col_15_R_t8 or th_col_15_G_t8 or th_col_15_Y_t8 or th_col_15_B_t8)
6 (th_col_16_R_t8 or th_col_16_G_t8 or th_col_16_Y_t8 or th_col_16_B_t8)
7 pos_16_p_14_16_t8 and tm_pos_16_p_14_16_t8 and th_pos_16_p_14_16_t8
8 act_6_t8 and tm_act_6_t8 and th_act_6_t8
```

Listing 12: Σ_m entries at time point 8

The planning process determines agent m 's actions. Actions performed by agent m and agent h are saved in the interface file: `/sdata/interfacefx.txt` with a flag equal to 1. These actions represent the Σ_{input} parameter as it is shown in the architecture in 4.3. Subsequently, and after the belief review process is executed, the flag indicator changes from 1 to 0. In Listing 13 we can see an example of the information stored in the interface file:

```
1 m:col_1_R_t1 and tm_col_1_R_t1 and (th_tm_col_1_G_t1 or th_tm_col_1_Y_t1 or
  th_tm_col_1_B_t1 or th_tm_col_1_R_t1):0
2 m:col_2_B_t2 and tm_col_2_B_t2 and (th_tm_col_2_G_t2 or th_tm_col_2_Y_t2 or
  th_tm_col_2_B_t2 or th_tm_col_2_R_t2):0
3 m:pos_3_p_11_12_t3 and tm_pos_3_p_11_12_t3 and th_pos_3_p_11_12_t3:0
4 m:act_1_t4 and tm_act_1_t4 and th_act_1_t4:0
5 h:(th_col_15_R_t5 or th_col_15_G_t5 or th_col_15_Y_t5 or th_col_15_B_t5):0
```

```

6 h:(th_col_16_R_t6 or th_col_16_G_t6 or th_col_16_Y_t6 or th_col_16_B_t6):0
7 h:pos_16_p_14_16_t7 and tm_pos_16_p_14_16_t7 and th_pos_16_p_14_16_t7:0
8 h:act_6_t8 and tm_act_6_t8 and th_act_6_t8:0

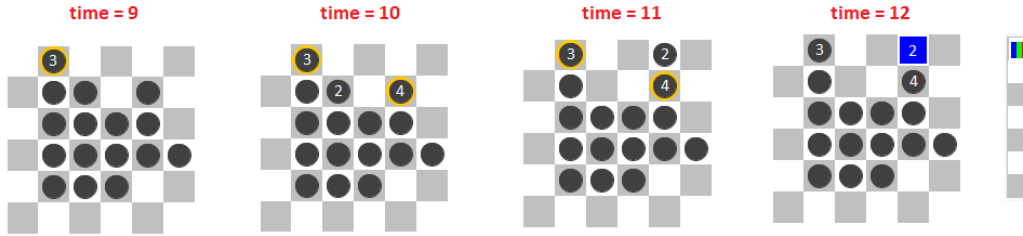
```

Listing 13: Interface file

As shown above, the interface file keeps the history of actions performed by the players, allowing us to follow the game’s evolution.

Grouping cards

During the game, agent m obtains new information that it will use to achieve the goal of grouping cards of the same color together in cooperation with agent h . As an example of m ’s ability to group cards based on previous observations, we can consider the following scenario illustrated in Figure 13.

Figure 13: Agent m ’s round 2

At time point 10, agent m observes that card 4 is blue and, thanks to a previous observation at time 2 (see Figure 11), agent m knows that card 2 is also blue, thereby m will try to achieve the goal of grouping these two cards.

The system calls the planning module with three parameters $\langle \Sigma, Op, \alpha_G \rangle$, as detailed in Definition 28.

The belief base Σ includes the SSA11 axiom as part of agent m ’s initial core beliefs (section 5.5.1). According to SSA11, if card x has a new position p at time t , the position of the other cards, the colors of all cards, and the status of all hints remain unchanged from $t - 1$ to t .

The formula used for implementing SSA11 is included in file `/sdata/00_axioms.touist` that contains the set of formulas representing agent m ’s core beliefs.

The set X_c used in algorithm 4 is generated by the system based on the observed cards. Next, we show the content of X_c for our example:

```

G|3;
Y|
R|1;
B|2;4;

```

The set X_c considers four rows, one per color, and is generated based on the observations performed by agent m . In addition, according to algorithm 4 for grouping cards, we also need to consider the set of deductions that agent m makes about the

possible color of a card based on the hints provided by the agent \mathfrak{h} . In algorithm 4, the system assigns the set of deductions to the variable H_c . Thus, in our example based on the fact that agent \mathfrak{h} still does not mark any card, the variable H_c set is empty.

We join the sets X_c and H_c in the set S , and we filter and sorted S according to algorithm 4 :

```
B|2;4;
R|1;
G|3;
```

Now that we have the set S sorted in descending order, agent \mathfrak{m} will try to group these cards starting from the first element in the list. Thus, the system extracts the first element from S and assigns it to X_c as is indicated in algorithm 4. In order to see if it is possible for agent \mathfrak{m} to group the cards in X_c , agent \mathfrak{m} evaluates $\mathcal{P}_g(\alpha_{groupCards_N(X_c)}^t, c)$ specified in sub-section 5.6.2 as it is shown in Listing 14.

```

1 $now = 10
2 $G2CARDS = [2, 4]
3 $G2COLOR = B
4 $X(B,2,1) = [[2, 4]]
5 $I = [pos_1_p_12_12, pos_2_p_12_13, pos_3_p_11_12, pos_4_p_12_15,
      pos_5_p_13_12, pos_6_p_13_13, pos_7_p_13_14, pos_8_p_13_15, pos_9_p_14_12,
      pos_10_p_14_13, pos_11_p_14_14, pos_12_p_14_15, pos_13_p_15_12,
      pos_14_p_15_13, pos_15_p_15_14, pos_16_p_14_16]
6 $POSINI(1) = [p_12_12]
7 $POSINI(2) = [p_12_13]
8 $POSINI(3) = [p_11_12]
9 $POSINI(4) = [p_12_15]
10 $POSINI(5) = [p_13_12]
11 $POSINI(6) = [p_13_13]
12 $POSINI(7) = [p_13_14]
13 $POSINI(8) = [p_13_15]
14 $POSINI(9) = [p_14_12]
15 $POSINI(10) = [p_14_13]
16 $POSINI(11) = [p_14_14]
17 $POSINI(12) = [p_14_15]
18 $POSINI(13) = [p_15_12]
19 $POSINI(14) = [p_15_13]
20 $POSINI(15) = [p_15_14]
21 $POSINI(16) = [p_14_16]
22 $NEIG(p_12_12) = [p_12_11]
23 $NEIG(p_12_13) = [p_11_13, p_12_14]
24 $NEIG(p_11_12) = [p_10_12, p_11_13, p_11_11]
25 $NEIG(p_12_15) = [p_11_15, p_12_16, p_12_14]
26 $NEIG(p_13_12) = [p_13_11]
27 $NEIG(p_13_13) = []
28 $NEIG(p_13_14) = [p_12_14]
29 $NEIG(p_13_15) = [p_13_16]
30 $NEIG(p_14_12) = [p_14_11]
31 $NEIG(p_14_13) = []
32 $NEIG(p_14_14) = []
33 $NEIG(p_14_15) = [p_15_15]
34 $NEIG(p_15_12) = [p_16_12, p_15_11]
35 $NEIG(p_15_13) = [p_16_13]
36 $NEIG(p_15_14) = [p_16_14, p_15_15]
37 $NEIG(p_14_16) = [p_13_16, p_15_16, p_14_17]
```

```

38 $NEIGBOR(p_12_12) = [p_11_12, p_13_12, p_12_13, p_12_11]
39 $NEIGBOR(p_12_13) = [p_11_13, p_13_13, p_12_14, p_12_12]
40 $NEIGBOR(p_11_12) = [p_10_12, p_12_12, p_11_13, p_11_11]
41 $NEIGBOR(p_12_15) = [p_11_15, p_13_15, p_12_16, p_12_14]
42 $NEIGBOR(p_13_12) = [p_12_12, p_14_12, p_13_13, p_13_11]
43 $NEIGBOR(p_13_13) = [p_12_13, p_14_13, p_13_14, p_13_12]
44 $NEIGBOR(p_13_14) = [p_12_14, p_14_14, p_13_15, p_13_13]
45 $NEIGBOR(p_13_15) = [p_12_15, p_14_15, p_13_16, p_13_14]
46 $NEIGBOR(p_14_12) = [p_13_12, p_15_12, p_14_13, p_14_11]
47 $NEIGBOR(p_14_13) = [p_13_13, p_15_13, p_14_14, p_14_12]
48 $NEIGBOR(p_14_14) = [p_13_14, p_15_14, p_14_15, p_14_13]
49 $NEIGBOR(p_14_15) = [p_13_15, p_15_15, p_14_16, p_14_14]
50 $NEIGBOR(p_15_12) = [p_14_12, p_16_12, p_15_13, p_15_11]
51 $NEIGBOR(p_15_13) = [p_14_13, p_16_13, p_15_14, p_15_12]
52 $NEIGBOR(p_15_14) = [p_14_14, p_16_14, p_15_15, p_15_13]
53 $NEIGBOR(p_14_16) = [p_13_16, p_15_16, p_14_17, p_14_15]
54 $HINTS = [1, 2, 3, 4, 5, 6, 7]
55 $AVHINTS = [1, 2]
56 $ACHINTS = []
57 $COLORS = [R,G,B,Y]
58 $OPOS($now) = [p_12_12, p_12_13, p_11_12, p_12_15, p_13_12, p_13_13, p_13_14,
    p_13_15, p_14_12, p_14_13, p_14_14, p_14_15, p_15_12, p_15_13, p_15_14,
    p_14_16]
59 $PERIM = [p_12_11, p_11_13, p_12_14, p_10_12, p_11_13, p_11_11, p_11_15,
    p_12_16, p_12_14, p_13_11, p_12_14, p_13_16, p_14_11, p_15_15, p_16_12,
    p_15_11, p_16_13, p_16_14, p_15_15, p_13_16, p_15_16, p_14_17]
60 $GRID = [p_11_12, p_12_12, p_13_12, p_14_12, p_15_12, p_15_13, p_15_14,
    p_14_15, p_14_16, p_13_15, p_12_15, p_13_14, p_13_13, p_12_13]
61 $CARDS = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
62
63 $t = $now + 1
64
65 bigand $x,$p in $G2CARDS, $POSINI($x):
66   pos($x,$p,$t)
67 end
68
69 ;; The cards in X are currently not neighboring cards
70 not (bigand $i, $X in [1..1], $X($G2COLOR,2,$i) :
71   (bigor $x, $p, $x1, $p1 in $X, $POSINI($x), $X, $POSINI($x1)
72     when ($p1 in $NEIGBOR($p)) and ($x != $x1):
73     (pos($x,$p,$t) and pos($x1,$p1,$t))
74   end)
75 end)
76
77 ;;At least 1 card in X has a neighboring free position in common
78 (bigor $X in $X($G2COLOR ,2,1) :
79   (bigor $x, $p, $x1, $p1 in $X, $POSINI($x), $X, $POSINI($x1)
80     when (card($NEIG($p))>=1 or card($NEIG($p1))>=1) and ($x != $x1) :
81     (pos($x,$p,$t) and pos($x1,$p1,$t))
82   end)
83 end)

```

Listing 14: Pre-condition for grouping 2 cards

If the precondition is SAT, this means that it is possible to group the two cards, so next the machine can use the goal : $\alpha^t_{groupCards_N(X_c)}$ represented by the following formula:

```

1 $now = 10
2 $G2CARDS = [2, 4]

```

```

3 $G2COLOR = B
4 $X(B,2,1) = [[2, 4]]
5 $I = [pos_1_p_12_12, pos_2_p_12_13, pos_3_p_11_12, pos_4_p_12_15,
      pos_5_p_13_12, pos_6_p_13_13, pos_7_p_13_14, pos_8_p_13_15, pos_9_p_14_12,
      pos_10_p_14_13, pos_11_p_14_14, pos_12_p_14_15, pos_13_p_15_12,
      pos_14_p_15_13, pos_15_p_15_14, pos_16_p_14_16]
6 $POSINI(1) = [p_12_12]
7 $POSINI(2) = [p_12_13]
8 $POSINI(3) = [p_11_12]
9 $POSINI(4) = [p_12_15]
10 $POSINI(5) = [p_13_12]
11 $POSINI(6) = [p_13_13]
12 $POSINI(7) = [p_13_14]
13 $POSINI(8) = [p_13_15]
14 $POSINI(9) = [p_14_12]
15 $POSINI(10) = [p_14_13]
16 $POSINI(11) = [p_14_14]
17 $POSINI(12) = [p_14_15]
18 $POSINI(13) = [p_15_12]
19 $POSINI(14) = [p_15_13]
20 $POSINI(15) = [p_15_14]
21 $POSINI(16) = [p_14_16]
22 $NEIG(p_12_12) = [p_12_11]
23 $NEIG(p_12_13) = [p_11_13, p_12_14]
24 $NEIG(p_11_12) = [p_10_12, p_11_13, p_11_11]
25 $NEIG(p_12_15) = [p_11_15, p_12_16, p_12_14]
26 $NEIG(p_13_12) = [p_13_11]
27 $NEIG(p_13_13) = []
28 $NEIG(p_13_14) = [p_12_14]
29 $NEIG(p_13_15) = [p_13_16]
30 $NEIG(p_14_12) = [p_14_11]
31 $NEIG(p_14_13) = []
32 $NEIG(p_14_14) = []
33 $NEIG(p_14_15) = [p_15_15]
34 $NEIG(p_15_12) = [p_16_12, p_15_11]
35 $NEIG(p_15_13) = [p_16_13]
36 $NEIG(p_15_14) = [p_16_14, p_15_15]
37 $NEIG(p_14_16) = [p_13_16, p_15_16, p_14_17]
38 $NEIGBOR(p_12_12) = [p_11_12, p_13_12, p_12_13, p_12_11]
39 $NEIGBOR(p_12_13) = [p_11_13, p_13_13, p_12_14, p_12_12]
40 $NEIGBOR(p_11_12) = [p_10_12, p_12_12, p_11_13, p_11_11]
41 $NEIGBOR(p_12_15) = [p_11_15, p_13_15, p_12_16, p_12_14]
42 $NEIGBOR(p_13_12) = [p_12_12, p_14_12, p_13_13, p_13_11]
43 $NEIGBOR(p_13_13) = [p_12_13, p_14_13, p_13_14, p_13_12]
44 $NEIGBOR(p_13_14) = [p_12_14, p_14_14, p_13_15, p_13_13]
45 $NEIGBOR(p_13_15) = [p_12_15, p_14_15, p_13_16, p_13_14]
46 $NEIGBOR(p_14_12) = [p_13_12, p_15_12, p_14_13, p_14_11]
47 $NEIGBOR(p_14_13) = [p_13_13, p_15_13, p_14_14, p_14_12]
48 $NEIGBOR(p_14_14) = [p_13_14, p_15_14, p_14_15, p_14_13]
49 $NEIGBOR(p_14_15) = [p_13_15, p_15_15, p_14_16, p_14_14]
50 $NEIGBOR(p_15_12) = [p_14_12, p_16_12, p_15_13, p_15_11]
51 $NEIGBOR(p_15_13) = [p_14_13, p_16_13, p_15_14, p_15_12]
52 $NEIGBOR(p_15_14) = [p_14_14, p_16_14, p_15_15, p_15_13]
53 $NEIGBOR(p_14_16) = [p_13_16, p_15_16, p_14_17, p_14_15]
54 $HINTS = [1, 2, 3, 4, 5, 6, 7]
55 $AVHINTS = [1, 3]
56 $ACHINTS = []
57 $COLORS = [R,G,B,Y]
58 $OPOS($now) = [p_12_12, p_12_13, p_11_12, p_12_15, p_13_12, p_13_13, p_13_14,
      p_13_15, p_14_12, p_14_13, p_14_14, p_14_15, p_15_12, p_15_13, p_15_14,
      p_14_16]
59 $PERIM = [p_12_11, p_11_13, p_12_14, p_10_12, p_11_13, p_11_11, p_11_15,

```



```

    p_12_16, p_12_14, p_13_11, p_12_14, p_13_16, p_14_11, p_15_15, p_16_12,
    p_15_11, p_16_13, p_16_14, p_15_15, p_13_16, p_15_16, p_14_17]
60 $GRID = [p_11_12, p_12_12, p_13_12, p_14_12, p_15_12, p_15_13, p_15_14,
    p_14_15, p_14_16, p_13_15, p_12_15, p_13_14, p_13_13, p_12_13]
61 $CARDS = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
62
63 ;;GOAL
64 $t = $now + 1
65 (bigor $X, $x1, $p1, $x2, $p2, $tp in
66     $X($G2COLOR ,2,1), $X, $POSINI($x1), $X diff [$x1], $POSINI($x2),
    $NEIG($p2) :
67     (pos($x1,$tp,$t) and pos($x2,$p2,$t))
68 end)
69 ;;END_GOAL

```

Listing 15: Goal formula for grouping two cards

Thereafter, the system generates the AST of the goal formula for grouping two cards using TouIST. The system saves the AST in a file which represents the α_G input to the planning module as it is shown in Figure 6.

```

$ ~/sw/code/yokai_master/sdata/goals$ ./touist.exe Goal_32.touist --show
(pos(2,p_11_15,11) and pos(4,p_12_15,11) or
(pos(2,p_12_14,11) and pos(4,p_12_15,11) or
(pos(2,p_12_16,11) and pos(4,p_12_15,11) or
(pos(4,p_11_13,11) and pos(2,p_12_13,11) or
(pos(4,p_12_14,11) and pos(2,p_12_13,11)

```

The next step is to call the planning module. The planner will try to find an action from the set of actions Op that will allow to achieve the goal α_G starting from the initial state Σ . At this point in our example, agent m 's beliefs contained in Σ are indexed at time equal to 10. Thus, the planning module will use information provided by the successor state axiom SS11, specified in Section 5.5.1, for deducing what will be true at time 11, which is index time for α_G .

Action selection

Afterwards the system runs the planning module with the three input parameters shown in Figure 6. During the planning process, the system enters in a loop generating formulas containing candidate plans until one of the formulas is UnSAT. If this is the case, then the formula contains a valid plan. In Listing 16, we show an example of a planning formula containing a candidate plan for grouping two cards moving card 2 to position `p_11_15` at slot time 11.

```

1 (not ([m](
2 not tm_col_5_G_t10 and
3 not tm_col_6_Y_t10 and
4 .
5 .
6 col_1_R_t10 and tm_col_1_R_t10 and (th_tm_col_1_G_t10 or th_tm_col_1_Y_t10 or
    th_tm_col_1_B_t10 or th_tm_col_1_R_t10) and
7 col_2_B_t10 and tm_col_2_B_t10 and (th_tm_col_2_G_t10 or th_tm_col_2_Y_t10 or
    th_tm_col_2_B_t10 or th_tm_col_2_R_t10) and
8 act_1_t10 and tm_act_1_t10 and th_act_1_t10 and

```

```

9 (th_col_15_R_t10 or th_col_15_G_t10 or th_col_15_Y_t10 or th_col_15_B_t10)
  and
10 (th_col_16_R_t10 or th_col_16_G_t10 or th_col_16_Y_t10 or th_col_16_B_t10)
  and
11 pos_3_p_11_12_t10 and tm_pos_3_p_11_12_t10 and th_pos_3_p_11_12_t10 and
12 pos_16_p_14_16_t10 and tm_pos_16_p_14_16_t10 and th_pos_16_p_14_16_t10 and
13 act_2_t10 and tm_act_2_t10 and th_act_2_t10 and
14 col_3_G_t10 and tm_col_3_G_t10 and (th_tm_col_3_G_t10 or th_tm_col_3_Y_t10 or
  th_tm_col_3_B_t10 or th_tm_col_3_R_t10) and
15 col_4_B_t10 and tm_col_4_B_t10 and (th_tm_col_4_G_t10 or th_tm_col_4_Y_t10 or
  th_tm_col_4_B_t10 or th_tm_col_4_R_t10) and
16
17 ;; Current Positions
18 pos_1_p_12_12_t10 and
19 pos_2_p_12_13_t10 and
20 pos_4_p_12_15_t10 and
21 pos_5_p_13_12_t10 and
22 pos_6_p_13_13_t10 and
23 pos_7_p_13_14_t10 and
24 pos_8_p_13_15_t10 and
25 pos_9_p_14_12_t10 and
26 pos_10_p_14_13_t10 and
27 pos_11_p_14_14_t10 and
28 pos_12_p_14_15_t10 and
29 pos_13_p_15_12_t10 and
30 pos_14_p_15_13_t10 and
31 pos_15_p_15_14_t10 and
32 pos_3_p_11_12_t10 and tm_pos_3_p_11_12_t10 and th_pos_3_p_11_12_t10 and
33 pos_16_p_14_16_t10 and tm_pos_16_p_14_16_t10 and th_pos_16_p_14_16_t10 and
34 .
35 .
36 ;;SSA11
37 ((pos_2_p_11_15_t11 and not pos_2_p_11_15_t10) =>
38 ((pos_1_p_12_12_t11 <=> pos_1_p_12_12_t10) and
39 ((pos_3_p_11_12_t11 <=> pos_3_p_11_12_t10) and
40 ((pos_4_p_12_15_t11 <=> pos_4_p_12_15_t10) and
41 ((pos_5_p_13_12_t11 <=> pos_5_p_13_12_t10) and
42 ((pos_6_p_13_13_t11 <=> pos_6_p_13_13_t10) and
43 ((pos_7_p_13_14_t11 <=> pos_7_p_13_14_t10) and
44 ((pos_8_p_13_15_t11 <=> pos_8_p_13_15_t10) and
45 ((pos_9_p_14_12_t11 <=> pos_9_p_14_12_t10) and
46 ((pos_10_p_14_13_t11 <=> pos_10_p_14_13_t10) and
47 ((pos_11_p_14_14_t11 <=> pos_11_p_14_14_t10) and
48 ((pos_12_p_14_15_t11 <=> pos_12_p_14_15_t10) and
49 ((pos_13_p_15_12_t11 <=> pos_13_p_15_12_t10) and
50 ((pos_14_p_15_13_t11 <=> pos_14_p_15_13_t10) and
51 ((pos_15_p_15_14_t11 <=> pos_15_p_15_14_t10) and
52 (pos_16_p_14_16_t11 <=> pos_16_p_14_16_t10)))))))))) and
53 .
54 .
55 )=> ([m]now_t10 and [m] (not mark_2_1_t10 and not mark_2_2_t10 and not
  mark_2_3_t10 and not mark_2_4_t10 and not mark_2_5_t10 and not
  mark_2_6_t10 and not mark_2_7_t10) and plus(pos_2_p_11_15_t11 and
  tm_pos_2_p_11_15_t11 and th_pos_2_p_11_15_t11, [m]((pos_2_p_11_15_t11 and
  pos_4_p_12_15_t11) or (pos_2_p_12_14_t11 and pos_4_p_12_15_t11) or (
  pos_2_p_12_16_t11 and pos_4_p_12_15_t11) or (pos_4_p_11_13_t11 and
  pos_2_p_12_13_t11) or (pos_4_p_12_14_t11 and pos_2_p_12_13_t11))))))

```

Listing 16: Planning formula for moving cards

The planning formula expressed in $\mathcal{L}^+(Atm)$ is firstly transformed into its equiv-

alent propositional logic formula and secondly it is sent to the SAT solver TouIST for checking satisfiability. When the planning module finds a valid plan, then the action is inserted in the interface file with flag equal to one, as it is shown in Listing 17:

```

1 m:col_1_R_t1 and tm_col_1_R_t1 and (th_tm_col_1_G_t1 or th_tm_col_1_Y_t1 or
   th_tm_col_1_B_t1 or th_tm_col_1_R_t1):0
2 m:col_2_B_t2 and tm_col_2_B_t2 and (th_tm_col_2_G_t2 or th_tm_col_2_Y_t2 or
   th_tm_col_2_B_t2 or th_tm_col_2_R_t2):0
3 m:pos_3_p_11_12_t3 and tm_pos_3_p_11_12_t3 and th_pos_3_p_11_12_t3:0
4 m:act_1_t4 and tm_act_1_t4 and th_act_1_t4:0
5 h:(th_col_15_R_t5 or th_col_15_G_t5 or th_col_15_Y_t5 or th_col_15_B_t5):0
6 h:(th_col_16_R_t6 or th_col_16_G_t6 or th_col_16_Y_t6 or th_col_16_B_t6):0
7 h:pos_16_p_14_16_t7 and tm_pos_16_p_14_16_t7 and th_pos_16_p_14_16_t7:0
8 h:act_6_t8 and tm_act_6_t8 and th_act_6_t8:0
9 m:col_3_G_t9 and tm_col_3_G_t9 and (th_tm_col_3_G_t9 or th_tm_col_3_Y_t9 or
   th_tm_col_3_B_t9 or th_tm_col_3_R_t9):0
10 m:col_4_B_t10 and tm_col_4_B_t10 and (th_tm_col_4_G_t10 or th_tm_col_4_Y_t10
   or th_tm_col_4_B_t10 or th_tm_col_4_R_t10):0
11 m:pos_2_p_11_15_t11 and tm_pos_2_p_11_15_t11 and th_pos_2_p_11_15_t11:1

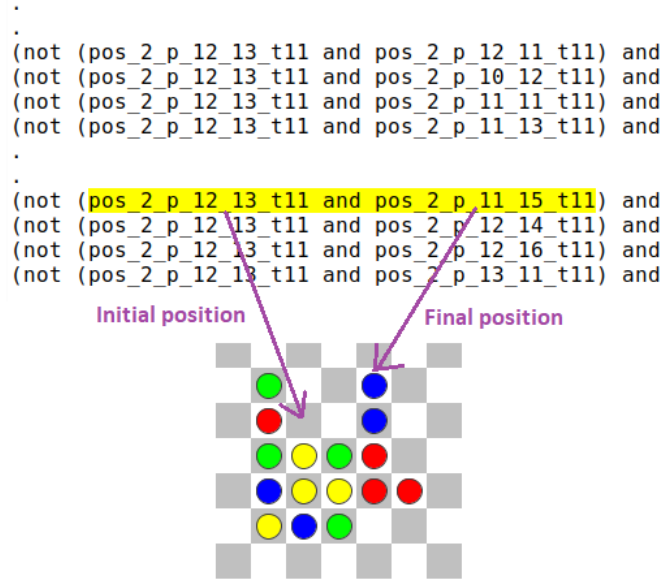
```

Listing 17: Interface file after the planning process finish

Belief revision

At this stage, we can say that the planning process has finished its task and now it is time to insert the new action Σ_{input} into the belief base. The belief revision module will be in charge of this work.

Since Σ_{input} consists of moving card 2 to a new position, the belief revision process will remove the previous belief containing the old position for this card and insert the new action in Σ_m . The belief revision module will revise the belief base using the integrity constraint ICP2 contained in Σ_c , which states that a given card can not be in two different positions at the same time. Thus, the belief revision module will first call the *belief update* function detailed in Section 5.4.2.1 to increment of one unit the time indexes of all formulas contained in Σ_c . Thereafter and thanks to ICP2, the belief revision will detect an inconsistency between Σ_m and Σ_{input} , consequently it will choose to remove the old belief from Σ_m and insert the new Σ_{input} .

Integrity constraint ICP2Figure 14: ICP2 in Σ_c

Given that now we have the following pieces of information in Σ_m when the belief revision module tries to save the new Σ_{input} in Σ_m , it will detect an inconsistency:

```

1 .
2 .
3 pos_1_p_12_12_t11
4 pos_2_p_12_13_t11
5 pos_4_p_12_15_t11
6 pos_5_p_13_12_t11
7 pos_6_p_13_13_t11
8 pos_7_p_13_14_t11
9 pos_8_p_13_15_t11
10 pos_9_p_14_12_t11
11 pos_10_p_14_13_t11
12 pos_11_p_14_14_t11
13 pos_12_p_14_15_t11
14 pos_13_p_15_12_t11
15 pos_14_p_15_13_t11
16 pos_15_p_15_14_t11
17 pos_3_p_11_12_t11 and tm_pos_3_p_11_12_t11 and th_pos_3_p_11_12_t11
18 pos_16_p_14_16_t11 and tm_pos_16_p_14_16_t11 and th_pos_16_p_14_16_t11
19 .
20 .

```

Listing 18: New Σ_m after performed the belief update process

Accordingly, to guarantee the consistency of the belief base, the belief revision module will remove the previous information related to the card 2 position and expand the belief base with the new belief. The section of Σ_m containing this information is shown next:

```

1 .

```

```

2 .
3 pos_1_p_12_12_t11
4 pos_4_p_12_15_t11
5 pos_5_p_13_12_t11
6 pos_6_p_13_13_t11
7 pos_7_p_13_14_t11
8 pos_8_p_13_15_t11
9 pos_9_p_14_12_t11
10 pos_10_p_14_13_t11
11 pos_11_p_14_14_t11
12 pos_12_p_14_15_t11
13 pos_13_p_15_12_t11
14 pos_14_p_15_13_t11
15 pos_15_p_15_14_t11
16 pos_3_p_11_12_t11 and tm_pos_3_p_11_12_t11 and th_pos_3_p_11_12_t11
17 pos_16_p_14_16_t11 and tm_pos_16_p_14_16_t11 and th_pos_16_p_14_16_t11
18 pos_2_p_11_15_t11 and tm_pos_2_p_11_15_t11 and th_pos_2_p_11_15_t11
19 .
20 .

```

Listing 19: New Σ_m after performed the belief revision process

Hierarchy of goals

In the following example we detail the steps performed by agent m for grouping cards based on the colors of the cards she observed in previous rounds. In Figure 15 we present the set of cards S known by agent m as it is indicated in Algorithm 4.

Agent m reads the first element in the S and tries to group 3 blue cards (2, 4 and 14), but it is not possible to group them because the precondition states that the cards should not be neighbors. In other words they must not be already grouped. Hence, given that the precondition for the first sub-set of cards is unsatisfiable, agent m tries with the second element in S and intends to group two red cards (1 and 8). Again in this case, the precondition is unsatisfiable because any movement combination of these two red cards will lead to an illegal move. Consequently, agent m will try with the third element in the hierarchy of goals S and intends to group two green cards (3 and 5). In this case, the precondition is satisfiable and after that the system calls the planning module in order to find an action to group cards 3 and 5. In our example, the planning module found a plan containing an action for moving card 3 close to card 5, as it is shown in Figure 15.

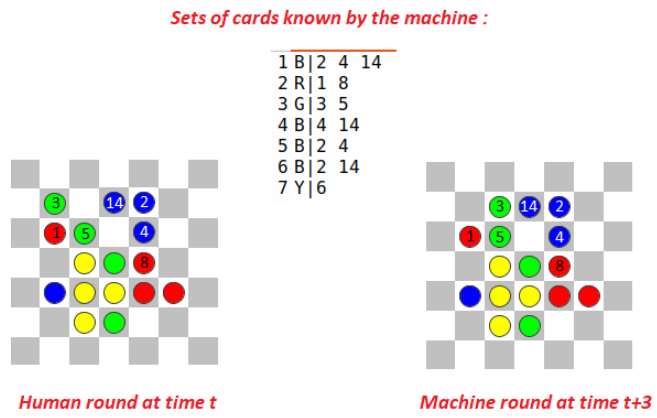


Figure 15: Hierarchy of goals

Bibliography

- [Amgoud *et al.* 2000] L. Amgoud, N. Maudet and S. Parsons. *Modelling dialogues using argumentation*. In Proceedings of the Fourth International Conference on MultiAgent Systems, pages 31–38. IEEE, 2000.
- [Aucher & Bolander 2013] G. Aucher and T. Bolander. *Undecidability in epistemic planning*. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), pages 27–33. AAAI Press, 2013.
- [Aucher & Schwarzentruher 2013] G. Aucher and F. Schwarzentruher. *On the complexity of dynamic epistemic logic*. In Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2013), 2013.
- [Aucher 2012] Guillaume Aucher. *Private announcement and belief expansion: an internal perspective*. Journal of Logic and Computation, vol. 22, no. 3, pages 451–479, 2012.
- [Audemard & Simon 2009] Gilles Audemard and Laurent Simon. *Predicting Learnt Clauses Quality in Modern SAT Solvers*. In Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI’09, page 399–404, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.
- [Audi 1973] R. Audi. *Intending*. The Journal of Philosophy, vol. 70, no. 13, pages 387–403, 1973.
- [Baltag *et al.* 1998] Alexandru Baltag, Lawrence S. Moss and Slawomir Solecki. *The Logic of Public Announcements, Common Knowledge, and Private Suspicions*. In Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge, TARK ’98, page 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- [Bard *et al.* 2020] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare and M. Bowling. *The Hanabi challenge: A new frontier for AI research*. Artificial Intelligence, vol. 280, 2020.
- [Bench-Capon 2003] T. J. M. Bench-Capon. *Persuasion in practical argument using value-based argumentation frameworks*. Journal of Logic and Computation, vol. 13(3), pages 429–448, 2003.
- [Bloem *et al.* 2018] Roderick Bloem, Nicolas Braud-Santoni, Vedad Hadzic, Uwe Egly, Florian Lonsing and Martina Seidl. *Expansion-Based QBF Solving Without Recursion*. CoRR, vol. abs/1807.08964, 2018.

- [Bolander & Andersen 2011] T. Bolander and M. B. Andersen. *Epistemic planning for single- and multi-agent systems*. Journal of Applied Non-Classical Logics, vol. 21, no. 1, pages 9–34, 2011.
- [Bolander *et al.* 2015a] T. Bolander, M. Holm Jensen and F. Schwarzenrüber. *Complexity Results in Epistemic Planning*. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), pages 2791–2797. AAAI Press, 2015.
- [Bolander *et al.* 2015b] T. Bolander, H. van Ditmarsch, A. Herzig, E. Lorini, P. Pardo and F. Schwarzenrüber. *Announcements to Attentive Agents*. Journal of Logic, Language and Information, vol. 25, no. 1, pages 1–35, 2015.
- [Bolander 2014] T. Bolander. *Seeing is believing: Formalising false-belief tasks in dynamic epistemic logic*. In A. Herzig and E. Lorini, editors, Proceedings of the European conference on Social Intelligence (ECSI-2014), pages 87–107, 2014.
- [Bonzon & Maudet 2011] E. Bonzon and N. Maudet. *On the Outcomes of Multiparty Persuasion*. In Proceedings of the 8th International Conference on Argumentation in Multi-Agent Systems (ArgMAS 2011), page 86–101. Springer-Verlag, 2011.
- [Brachman & Levesque 2004] Ronald Brachman and Hector Levesque. Knowledge representation and reasoning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [Bubeck & Büning 2007] Uwe Bubeck and Hans Kleine Büning. *Bounded Universal Expansion for Preprocessing QBF*. In SAT, 2007.
- [Budzyńska & Kacprzak 2008] K. Budzyńska and M. Kacprzak. *A Logic for Reasoning about Persuasion*. Fundamenta Informaticae, vol. 85, no. 1-4, pages 51–65, 2008.
- [Caridroit *et al.* 2017] T. Caridroit, J.-M. Lagniez, D. Le Berre, T. de Lima and V. Montmirail. *A SAT-Based Approach for Solving the Modal Logic S5-Satisfiability Problem*. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), pages 3864–3870. AAAI Press, 2017.
- [Cialdini 2001] R. B. Cialdini. Influence: science and practice. Allyn & Bacon, 2001.
- [Clarke *et al.* 2003] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu and Helmut Veith. *Counterexample-Guided Abstraction Refinement for Symbolic Model Checking*. J. ACM, vol. 50, no. 5, page 752–794, sep 2003.
- [Cooper *et al.* 2016] M. C. Cooper, A. Herzig, F. Maffre, F. Maris and P. Régnier. *A simple account of multi-agent epistemic planning*. In Proceedings of the 22nd

- European Conference on Artificial Intelligence (ECAI 2016), pages 193–201, 2016.
- [Cooper *et al.* 2021] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, Elise Perrotin and Pierre Régnier. *A lightweight epistemic logic and its application to planning*. Artificial Intelligence, vol. 298, page 103437, 2021.
- [Coste Marquis *et al.* 2006] Sylvie Coste Marquis, Daniel Le Berre, Florian Letombe and Pierre Marquis. *Complexity Results for Quantified Boolean Formulae Based on Complete Propositional Languages*. JSAT, vol. 1, pages 61–88, 03 2006.
- [Da Costa Pereira *et al.* 2011] C. Da Costa Pereira, A. Tettamanzi and S. Villata. *Changing One’s Mind: Erase or Rewind? Possibilistic Belief Revision with Fuzzy Argumentation Based on Trust*. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011), page 164–171. AAAI Press, 2011.
- [da Silva *et al.* 2018] Joana Galvão Gomes da Silva, David J Kavanagh, Tony Belpaeme, Lloyd Taylor, Konna Beeson, Jackie Andrade *et al.* *Experiences of a motivational interview delivered by a robot: qualitative study*. Journal of medical Internet research, vol. 20, no. 5, page e7737, 2018.
- [Davidson 1980] D. Davidson. *Essays on actions and events*. Clarendon Press, 1980.
- [Davis *et al.* 1962] Martin D. Davis, George Logemann and Donald W. Loveland. *A machine program for theorem-proving*. Commun. ACM, vol. 5, pages 394–397, 1962.
- [Dissing & Bolander 2020] L. Dissing and T. Bolander. *Implementing Theory of Mind on a Robot Using Dynamic Epistemic Logic*. In C. Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, (IJCAI 2020), pages 1615–1621, 2020.
- [Eger & Martens 2017] M. Eger and C. Martens. *Practical Specification of Belief Manipulation in Games*. In B. Magerko and J. P. Rowe, editors, Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17), pages 30–36. AAAI Press, 2017.
- [Eger *et al.* 2017] M. Eger, C. Martens and M. A. Cordoba. *An intentional AI for hanabi*. In IEEE Conference on Computational Intelligence and Games, CIG 2017, pages 68–75. IEEE, 2017.
- [Fagin *et al.* 1995] R. Fagin, J. Halpern, Y. Moses and M. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, 1995.
- [Fernandez *et al.* 2020] Jorge Fernandez, Olivier Gasquet, Andreas Herzig, Dominique Longin, Emiliano Lorini, Frédéric Maris and Pierre Régnier.

- TouIST: a Friendly Language for Propositional Logic and More*. In Christian Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 5240–5242. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Demos.
- [Gabbay & Guenther 2003] D. M. Gabbay and F. Guenther, editors. Modal epistemic and doxastic logic, pages 1–38. Springer Netherlands, Dordrecht, 2003.
- [Gerbrandy & Groeneveld 1997] J. Gerbrandy and W. Groeneveld. *Reasoning about information change*. Journal of Logic, Language, and Information, vol. 6, pages 147–196, 1997.
- [Ghallab *et al.* 2004] Malik Ghallab, Dana Nau and Paolo Traverso. Automated Planning: Theory and Practice. Elsevier - Morgan Kaufman, 2004.
- [Goldman 1979] A. Goldman. *What is Justified Belief?* In G. Pappas, editor, Justification and Knowledge, pages 1–25. D. Reidel, 1979.
- [Goldman 2006] A. I. Goldman. Simulating minds: The philosophy, psychology, and neuroscience of mindreading. Oxford University Press, 2006.
- [Goultiaeva & Bacchus 2010] Alexandra Goultiaeva and Fahiem Bacchus. *Exploiting QBF Duality on a Circuit Representation*. volume 1, 01 2010.
- [Halpern & Moses 1992] J. Y. Halpern and Y. Moses. *A guide to completeness and complexity for modal logics of knowledge and belief*. Artificial Intelligence, vol. 54, no. 3, pages 319–379, 1992.
- [Herzig & de Lima 2006] Andreas Herzig and Tiago de Lima. *Epistemic Actions and Ontic Actions: A Unified Logical Framework*. In IBERAMIA-SBIA, 2006.
- [Hunter 2015] A. Hunter. *Modelling the Persuadee in Asymmetric Argumentation Dialogues for Persuasion*. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI 2015), page 3055–3061. AAAI Press, 2015.
- [Janota *et al.* 2016] Mikoláš Janota, William Klieber, Joao Marques-Silva and Edmund Clarke. *Solving QBF with counterexample guided refinement*. Artificial Intelligence, vol. 234, pages 1–25, 2016.
- [Kanaoka & Mutlu 2015] Toshikazu Kanaoka and Bilge Mutlu. *Designing a motivational agent for behavior change in physical activity*. In Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, pages 1445–1450, 2015.
- [Klieber *et al.* 2010] William Klieber, Samir Sapra, Sicun Gao and Edmund M. Clarke. *A Non-prenex, Non-clausal QBF Solver with Game-State Learning*. In SAT, 2010.

- [Kominis & Geffner 2015] F. Kominis and H. Geffner. *Beliefs in multiagent planning: from one agent to many*. In Ronen I. Brafman, Carmel Domshlak, Patrik Haslum and Shlomo Zilberstein, editors, Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS 2015), pages 147–155. AAAI Press, 2015.
- [Krzywicki *et al.* 2016] Alfred Krzywicki, Wayne Wobcke, Michael Bain, John Calvo Martinez and Paul Compton. *Data mining for building knowledge bases: Techniques, architectures and applications*. The Knowledge Engineering Review, vol. -1, pages 1–27, 03 2016.
- [Ladner 1977] R. E. Ladner. *The Computational Complexity of Provability in Systems of Modal Propositional Logic*. SIAM Journal of Computing, vol. 6, no. 3, pages 467–480, 1977.
- [Lê Cong *et al.* 2018] S. Lê Cong, S. Pinchinat and F. Schwarzenrüber. *Small Undecidable Problems in Epistemic Planning*. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, pages 4780–4786. ijcai.org, 2018.
- [Levesque 1984] H. J. Levesque. *A logic of implicit and explicit belief*. In Proceedings of the Fourth AAAI Conference on Artificial Intelligence (AAAI'84), pages 198–202. AAAI Press, 1984.
- [Lisetti *et al.* 2013] Christine Lisetti, Reza Amini, Ugan Yasavur and Naphtali Rishé. *I can help you change! an empathic virtual agent delivers behavior change health interventions*. ACM Transactions on Management Information Systems (TMIS), vol. 4, no. 4, pages 1–28, 2013.
- [Lomuscio *et al.* 2017] A. Lomuscio, H. Qu and F. Raimondi. *MCMAS: an open-source model checker for the verification of multi-agent systems*. International Journal on Software Tools for Technology Transfer, vol. 19, pages 9–30, 2017.
- [Lorini & Romero 2019] E. Lorini and F. Romero. *Decision procedures for epistemic logic exploiting belief bases*. In Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), pages 944–952. IFAAMAS, 2019.
- [Lorini & Schwarzenrüber 2021] E. Lorini and F. Schwarzenrüber. *Multi-Agent Belief Base Revision*. In Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021). ijcai.org, 2021.
- [Lorini 2018] E. Lorini. *In Praise of Belief Bases: Doing Epistemic Logic Without Possible Worlds*. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pages 1915–1922. AAAI Press, 2018.
- [Lorini 2020] E. Lorini. *Rethinking epistemic logic with belief bases*. Artificial Intelligence, vol. 282, 2020.

- [Löwe *et al.* 2011] B. Löwe, E. Pacuit and A. Witzel. *DEL planning and some tractable cases*. In Proceedings of the 3rd International International Workshop on Logic, Rationality and Interaction (LORI 2011), pages 179–192. Springer Berlin Heidelberg, 2011.
- [Lundahl & Burke 2009] Brad Lundahl and Brian L Burke. *The effectiveness and applicability of motivational interviewing: A practice-friendly review of four meta-analyses*. Journal of clinical psychology, vol. 65, no. 11, pages 1232–1245, 2009.
- [Lutz 2006] C. Lutz. *Complexity and succinctness of public announcement logic*. In Proceedings of the Fifth international Joint Conference on Autonomous agents and Multiagent Systems, pages 137–143. ACM, 2006.
- [Makinson 1997] David Makinson. *Screened revision*. Theoria, vol. 63, pages 14–23, 1997.
- [Marques Silva & Sakallah 1996] J.P. Marques Silva and K.A. Sakallah. *GRASP—A new search algorithm for satisfiability*. In Proceedings of International Conference on Computer Aided Design, pages 220–227, 1996.
- [Miller & Rollnick 2012] William R Miller and Stephen Rollnick. *Motivational interviewing: Helping people change*. Guilford press, 2012.
- [Muisse *et al.* 2015] C. Muise, V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce and L. Sonenberg. *Planning over multi-agent epistemic states: A classical planning approach*. In Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015), pages 3327–3334. AAAI Press, 2015.
- [Muisse *et al.* 2021] C. Muise, V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce, and L. Sonenberg. *Efficient Multi-agent Epistemic Planning: Teaching Planners About Nested Belief*. Artificial Intelligence, vol. 302, 2021.
- [Olafsson *et al.* 2019] Stefan Olafsson, Teresa O’Leary and Timothy Bickmore. *Coerced change-talk with conversational agents promotes confidence in behavior change*. In Proceedings of the 13th EAI International Conference on Pervasive Computing Technologies for Healthcare, pages 31–40, 2019.
- [Perloff 2003] R. M. Perloff. *The dynamics of persuasion: Communication and attitudes in the 21st century*. L. Erlbaum, 2003.
- [Plaza 1989] J. A. Plaza. *Logics of public communications*. In M. Emrich, M. Pfeifer, M. Hadzikadic and Z. Ras, editors, Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems, 201–216, 1989.
- [Prakken 2006] H. Prakken. *Formal Systems for Persuasion Dialogue*. The Knowledge Engineering Review, vol. 21, no. 2, page 163–188, 2006.

- [Proietti & Yuste-Ginel 2019] C. Proietti and A. Yuste-Ginel. *Persuasive Argumentation and Epistemic Attitudes*. In Proceedings of the Second International Workshop on Dynamic Logic. New Trends and Applications (DALI 2019), volume 12005 of *LNCS*, pages 104–123. Springer-Verlag, 2019.
- [Rashotte 2009] L. Rashotte. *Social Influence*. In G. Ritzer and J. M. Ryan, editors, Concise Blackwell Encyclopedia of Sociology. Blackwell, 2009.
- [Schulman *et al.* 2011] Daniel Schulman, Timothy Bickmore and Candace Sidner. *An intelligent conversational agent for promoting long-term health behavior change using motivational interviewing*. In 2011 AAAI Spring Symposium Series, 2011.
- [Searle 1969] J. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University Press, Cambridge, 1969.
- [Shoham 2009] Y. Shoham. *Logical Theories of Intention and the Database Perspective*. *Journal of Philosophical Logic*, vol. 38, no. 6, pages 633–648, 2009.
- [Stalnaker 2002] R. Stalnaker. *Common ground*. *Linguistics and Philosophy*, vol. 25(5-6), pages 701–721, 2002.
- [Sörensson & Een 2005] Niklas Sörensson and Niklas Een. *Minisat v1.13-a SAT solver with conflict-clause minimization*. International Conference on Theory and Applications of Satisfiability Testing, 01 2005.
- [van Benthem 2003] Johan van Benthem. *Logic and the Dynamics of Information*. *Minds and Machines*, vol. 13, no. 4, pages 503–519, 2003.
- [van Ditmarsch *et al.* 2007] H. van Ditmarsch, W. van der Hoek and B. Kooi. *Dynamic epistemic logic*. Synthese Library. Springer Netherlands, 2007.
- [Walton & Krabbe 1995] D. Walton and E.C.W. Krabbe. *Commitment in dialogue: Basic concepts of interpersonal reasoning*. SUNY Series in Logic and Language. State University of New York Press, 1995.