Knowledge-Based Policies for Qualitative Decentralized POMDPs
Abdallah Saffidine, François Schwarzentruber, Bruno Zanuttini

To cite this version:
Abdallah Saffidine, François Schwarzentruber, Bruno Zanuttini. Knowledge-Based Policies for Qualitative Decentralized POMDPs. 32nd AAAI Conference on Artificial Intelligence, Feb 2018, New Orleans, United States. <hal-01646207>

HAL Id: hal-01646207
https://hal.archives-ouvertes.fr/hal-01646207
Submitted on 6 Dec 2017

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.
Knowledge-Based Policies for Qualitative Decentralized POMDPs

Abdallah Saffidine  
University of New South Wales, Sydney  
Australia

François Schwarzentruber  
Univ. Rennes, CNRS, IRISA  
France

Bruno Zanuttini  
UNICAEN, ENSICAEN, CNRS, GREYC  
14000 Caen, France

Abstract

Qualitative Decentralized Partially Observable Markov Decision Problems (QDec-POMDPs) constitute a very general class of decision problems. They involve multiple agents, decentralized execution, sequential decision, partial observability, and uncertainty. Typically, joint policies, which prescribe to each agent an action to take depending on its full history of (local) actions and observations, are huge, which makes it difficult to store them onboard, at execution time, and also hampers the computation of joint plans.

We propose and investigate a new representation for joint policies in QDec-POMDPs, which we call Multi-Agent Knowledge-Based Programs (MAKBPs), and which uses epistemic logic for compactly representing conditions on histories. Contrary to standard representations, executing an MAKBP requires reasoning at execution time, but we show that MAKBPs can be exponentially more succinct than any reactive representation.

Introduction

Knowledge-Based Programs (KBPs) (Fagin et al. 2003) describe policies that agents should perform according to their knowledge, such as ‘if $K_i \Phi$ then $\kappa$’ where $K_i$ is the knowledge modality for agent $i$, $\Phi$ is a formula, and $\kappa$ is a program. As shown by Lang and Zanuttini (2012; 2013) for the single-agent case, KBPs offer compact representations of policies for planning problems. Moreover, arguably, they provide a natural level of expressivity for experts to design policies for autonomous agents.

We are interested here in multi-agent decision problems, for which we generalize KBPs. Precisely, we consider Qualitative Decentralized Partially Observable Markov Decision Problems (QDec-POMDPs), as formalized by Brafman, Shani, and Zilberstein (2013). Such problems involve multiple agents, decentralized execution, sequential decision, partial observability, and uncertainty.

A single agent executing a policy for such a problem must reason about what the other agents are executing, what they have observed, and what they believe or know. However, this is taken into account only indirectly by standard representations of policies. A drawback of this is that such policies are typically huge. By expliciting the reasoning about other agents’ knowledge in Multi-Agent KBPs (MAKBPs), we provide more succinct representations. The counterpart is that executing MAKBPs requires reasoning, hence they do not constitute reactive policies. However, in many scenarios some reasoning at execution time may be affordable. Typical examples are applications in which executing atomic actions is not instantaneous (like in extraterraneous missions), so that deliberation can be performed in parallel.

Knowledge-based policies have been considered in other planning settings than that of QDec-POMDPs, in particular, they arise naturally in epistemic planning (Bolander and Andersen 2011). Other works have taken inspiration from reasoning about knowledge to design approaches for multi-agent (classical) planning (Koninis and Geffner 2015).

However, the originality of the (qualitative) Dec-PODMP model is that it assumes decentralized execution, but centralized planning. This makes it a challenging problem, as it requires to be able to plan offline a joint plan which can be executed in a decentralized fashion. The challenge is visible in the large body of work about the resolution of quantitative (stochastic) Dec-POMDPs (see Oliehoek and Amato (2016), Dibangoye et al. (2016) for instance).

Centralized planning with decentralized execution means that agents agree on a policy offline, then execute it independently of each other. Hanabi is a flagship example of such settings. In this cooperative card game with partial observability and limited communication, players typically agree on conventions before the game takes places. Researchers have started building computer players for Hanabi, notably by encoding human-designed policies (van den Bergh 2015; Cox et al. 2015). Furthermore, describing and using policies that involve epistemic reasoning to determine the next action is seen as a gateway to high-performance automated players (Osawa 2015). Yet, no formal frameworks allowing such policies have been put forward.

Example 1 (running) As a running example, assume Alice wants to meet Bob. She may take a train or a flight from her place to his, but there is a risk of air crew strike. Alice and Bob may call each other beforehand (centralized planning phase) to agree on the following plan.

Alice tries to fly. If there is a strike, she takes the train and listens to the radio to know whether Bob can know this. If the strike is not announced on the radio (hence Bob cannot know), after arriving at the station she will reach the airport to meet Bob there. Otherwise they will meet at her place of arrival. It can be seen that this plan, once agreed upon, can...
be executed successfully in a decentralized fashion.

Besides defining MAKBPs as a new representation of joint policies for QDec-POMDPs, we make the following contributions. We prove that this representation is as succinct as standard ones, e.g., joint policies trees, and we give families of problems for which it is exponentially more succinct. Contrary to standard representations, determining the action to execute next is nontrivial in MAKBPs, as it involves explicit reasoning about other agents’ knowledge. We then investigate the complexity of this execution problem and that of verifying whether a given MAKBP solves a given QDec-POMDP, and we show that both these essential problems are intractable: both are PSPACE-complete if the horizon is bounded, and verification is undecidable otherwise.¹

### Decentralized POMDPs

We define the planning tasks which we address in two steps; first the model, encoding the dynamics of actions and observations (Definition 2), then the planning task itself, which specifies an initial (belief) state, a goal, and a planning horizon (Definition 3). Our definitions essentially follow those given by Brafman, Shani, and Zilberstein (2013). We generalize them slightly by (1) allowing nondeterminism in the transitions and observations, (2) allowing the observations to depend on the previous state as well as the action and resulting state, and (3) allowing arbitrary observations rather than only the truth value of propositions.

For simplicity, we assume that all agents have the same actions, and do not consider explicit preconditions.²

**Definition 2 (factored QDec-POMDP model)** A factored model for a Qualitative Decentralized Partially Observable Markov Decision Problem (QDec-POMDP) is a tuple \( M = \langle I, X, A, \Omega, T \rangle \), where \( I, X, A, \Omega \) are finite set of agents, propositions, actions, and observations, respectively, and \( T \) is a transition function, which maps each couple \((s, a) \in 2^X \times A^I\) to a set of couples \(\{ (s', \omega) \} \subseteq 2^X \times \Omega^I\).

A Boolean assignment \( s \rightarrow X \) is called a state; \( s[x] \) denotes the value it assigns to \( x \). An element of \( A^I \) is called a joint action, and specifies one (individual) action \( a_i \) per agent \( i \). We use similar notation for observations. The transition function encodes the dynamics of actions and observations: \((s', \omega) \in T(s, a)\) means that when the state is \( s \) and the agents take joint action \( a \), it is possible that both the environment transitions to state \( s' \) and each agent \( i \) receives the observation \( \omega_i \). Note that the next states and observations depend on the current state and the joint action taken.

At execution time (when agents act), at any timestep \( t \) the environment is in some state \( s^t \in 2^X \). Each agent \( i \) (simultaneously and independently) chooses and takes (an individual) action \( a_i^t \in A^I \) defining the joint action \( a^t \). Then the environment nondeterministically chooses a couple \((s^{t+1}, \omega^t)\) from \( T(s^t, a^t) \); it transitions to state \( s^{t+1} \), each agent \( i \) receives the individual observation \( \omega_i^t \); and the process repeats.

**Definition 3 (factored QDec-POMDP)** A factored QDec-POMDP is a quadruple \( \Pi = \langle M, B^0, g, H \rangle \), where \( M = \langle I, X, A, \Omega, T \rangle \) is a factored QDec-POMDP model, \( B^0 \subseteq 2^X \) is the initial belief state, \( g \) is a Boolean formula over \( X \) called goal, and \( H \in \mathbb{N} \cup \{ \infty \} \) is the horizon.³

### Histories and policies

Let \( M = \langle I, X, A, \Omega, T \rangle \) be a QDec-POMDP model. A t-history for \( M \) is a sequence \( h^t = s^0a^0\omega^0s^1a^1\omega^1 \cdots s^{t-1}a^{t-1}\omega^{t-1}s^t \) satisfying \( \forall t < t', (s^{t+1}, \omega^t) \subseteq T(s^t, a^t) \). Of \( h^t \), the only information available to an agent \( i \in I \) is its local history \( h_i^t = a^0_i\omega^0_i a^1_i\omega^1_i \cdots a^{t-1}_i\omega^{t-1}_i \), that is, the sequence of (individual) actions it has taken and (individual) observations it has received. We write \( H^t \) for the set of all t(histories for \( M \) will always be clear from the context), and \( H^t \) for the set of all local t(histories for \( i \). We also use notation \( H^t \subseteq \bigcup_{t=0}^T H^t \) and \( H^t = \bigcup_{t=0}^T H^t \).

A (deterministic) t-policy \( \pi \) for \( i \) is a mapping \( \pi_i : H^t_i \rightarrow A \). Action \( \pi_i(h_i^t) \) is the one which \( i \) should take when its local history so far is \( h_i^t \). A joint t-policy \( \pi \) is simply a vector of t-policies, one for each agent \( i \). A history \( h^t = s^0a^0\omega^0 \cdots s^{t-1}a^{t-1}\omega^{t-1}s^t \) is said to be consistent with \( \pi \) if at all timesteps, each agent acts according to its policy, namely, for all timesteps \( t' < t - 1 \) and all agents \( i \), \( a_i^{t'+1} = \pi_i(a^0_i\omega^0_i \cdots a_i^{t'+1}\omega_i^{t'+1}) \).

A joint t-policy \( \pi \) is valid for \( \Pi = \langle M, B^0, g, H \rangle \) if for all \( H \)-histories \( s^0a^0\omega^0 \cdots s^{H-1}a^{H-1}\omega^{H-1}s^H \) with \( s^0 \in B^0 \) and consistent with \( \pi \), the final state \( s^H \) satisfies the goal \( g \). Note that we require the final state to satisfy \( g \); this implicitly requires that the agents know when \( g \) is achieved and maintain it until timestep \( H \) (e.g., using void actions if available). Finally, two joint policies are said to be t-equivalent (wrt \( \Pi \)) if they induce exactly the same sets of consistent t-histories.

In this paper, we are interested in joint policies which are computed offline in a centralized manner (as opposed to execution, which is decentralized), as is standard in the literature on Dec-POMDPs (Brafman, Shani, and Zilberstein 2013; Oliehoek and Amato 2016; Dibangoye et al. 2016).

### Representations

Designing compact representations for (joint) policies is crucial, due to the combinatorial nature of \( H_i^t \). The most direct representations are (1) by policy trees, which the agent follows by branching on the observations received and executing the actions stored at the nodes, and (2) by finite state controllers, i.e., labelled automata in which transitions are fired by observations and each state holds an action to take (see Kumar, Mostafa, and Zilberstein (2016)).⁴

¹A simple demonstration of executions of KBPs can be found in the tool Hintikka’s world (http://people.ii.uni.louisiana.edu/ Francois.Schwarzenbruber/hintikkasworld/).

²The trick to model unexecutable joint actions consists of adding a sink state to which any unexecutable joint action leads.

³By assuming \( g \) to be a Boolean formula, we stick to the standard definition of a Dec-POMDP. However, it is easily seen that allowing epistemic goals, e.g., that all agents know the value of some variable, would not change the results in the paper.

⁴Note that finite state controllers constitute a much more general class of representations than policy trees. In particular, if there
\[ w: \{x\} \xrightarrow{1} w': \emptyset \xrightarrow{2} w'': \{x\} \]

Figure 1: Example of a Kripke structure (each world \( w \) is annotated with its valuation \( V(w) \))

A well-known drawback of such representations is that they can be huge even for simple problems. Observe indeed that there are as many as \(|\Omega|\)\(^t\) local \( t \)-histories, each of which can yield a distinct path in the tree or automaton. On the other hand, an advantage of using these representations is that they are reactive, in the sense that they are easily executable. Taking inspiration from Bäckström and Jonsson (2012), we formalize this as follows.

**Definition 4 (execution problem)** The execution problem takes as input a QDec-POMDP model \( M \), an initial belief state \( B_0 \), a joint policy \( \pi \), an agent \( i \), and a local history \( h^i_t \) such that there is a history \( h_t^i \) consistent with \( \pi \) and inducing \( h^i_t \). It asks what action \( \pi(h^i_t) \) is, that is, what action \( \pi \) prescribes to \( i \) at timestep \( t + 1 \).

**Definition 5 (reactive policy representation)** A class \( R \) of representations of policies is said to be reactive if the execution problem restricted to policies represented in \( R \) is solvable in polynomial time.

Clearly, policy trees and finite state controllers constitute reactive representations: executing one consists of iteratively changing state according to the observation just received and taking the action found there. The rest of the paper provides a proper generalization of them, which is not reactive but has the essential advantage of being more succinct.

**Background on Epistemic Logic**

Let \( X \) be a set of variables and \( I \) be a set of agents. A Kripke structure represents an epistemic situation, that is, a description of the state of the variables and of what the agents know about it, about what the other agents know about it, etc. (Hintikka 1962). This is done through a graph of possible worlds and indistinguishability relations \( \sim_i \) (for each agent \( i \)). Intuitively, a possible world \( w \) represents a possible state over \( X \) (either the actual one, or one which an agent imagines) together with the epistemic situation for all agents, assuming that \( w \) is the actual world. Two such worlds are related by \( \sim_i \) if whenever the actual world is one of them, agent \( i \) thinks it might as well be the other one. Note that we model knowledge rather than beliefs (which might be false); precisely, we rely on an \( S5_n \) semantics, and in particular, an agent may know a formula \( \Phi \) only if \( \Phi \) is true.

**Definition 6 (Kripke structure)** A Kripke structure over \( X \) is a tuple \( S = \langle W, (\sim_i)_{i \in I}, V \rangle \), where \( W \) is nonempty set of worlds, \( \sim_i \) is an equivalence relation over \( W \) for all agents \( i \in I \), and \( V: W \to 2^X \) is a valuation function, which associates an assignment to \( X \) to each world in \( W \).

are (feasible) loops in the automaton, the policy can be executed for an infinite number of steps.

Figure 1 shows a Kripke structure with three worlds \( w, w', w'' \) (reflexive links are omitted for simplicity). Variable \( x \) is true in \( w \) and \( w'' \), and false in \( w' \). Note that different worlds may have the same valuation. The \( \sim \)-equivalence classes are \( \{w, w'\} \) and \( \{w''\} \). Hence when the actual situation is \( w \), agent 1 thinks it might be \( w' \) instead, and hence, it thinks that 2 may think that the actual world is \( w'' \). Contrastingly, when the actual state is \( w \) or \( w' \), agent 1 knows that the actual state is not \( w'' \). It can also be seen that when the actual world is \( w \), agent 1 does not know the value of \( x \), while it knows that \( x \) is true when the actual world is \( w'' \).

Kripke structures are essentially used for giving semantics to epistemic formulas (over \( X \) and \( I \)), which are logical formulas \( \Phi \) generated by the following grammar:

\[
\Phi ::= x \mid \neg \Phi \mid \Phi \lor \Phi \mid K_i \Phi \mid KW_i \Phi
\]

where \( x \) ranges over \( X \), \( i \) over \( I \), \( K_i \Phi \) is read “agent \( i \) knows that \( \Phi \) holds”, and \( KW_i \Phi \) is read “agent \( i \) knows whether \( \Phi \) holds or not”. An epistemic formula \( \Phi \) is said to be subjective for agent \( i \) if all propositional variables \( x \) are in the scope of a \( K_i \) or \( KW_i \) modality. This means that \( \Phi \) refers to the knowledge of \( i \) only; in particular, \( i \) is able to evaluate such a formula, while in general it does not know enough of the actual world to evaluate an objective (propositional) formula about it. Finally, the epistemic depth \( d(\Phi) \) of \( \Phi \) is the deepest nesting of modalities in \( \Phi \).

The truth condition \( S, w \models \Phi \) is read “\( S, w \) satisfies \( \Phi \)”, and means that \( \Phi \) is true at world \( w \) in the Kripke structure \( S \). It is defined by induction on \( \Phi \), with the obvious definition for Boolean connectives:

- \( S, w \models x \) if \( x \) is assigned to true by \( V(w) \);
- \( S, w \models K_i \Phi \) if \( \forall w', (w \sim_i w' \Rightarrow S, w' \models \Phi) \) holds;
- \( S, w \models KW_i \Phi \) if \( S, w \models K_i \Phi \) or \( S, w \models K_i \neg \Phi \).

We insist that despite the fact that the actual world \( w \) is given in the definition, in general the agents do not know what the actual world is. For instance, on Figure 1 we have \( S, w \models \neg K_1 x \land K_1 (x \lor \neg KW_2 x) \). In this paper, we use epistemic formulas as branching conditions in policies, and Kripke structures for providing them with operational semantics. In these structures, possible worlds essentially correspond to histories for the QDec-POMDP model at hand, and two histories are indistinguishable of each other for an agent \( i \) if they induce the same local history for \( i \).

**Multi-Agent Knowledge-Based Programs**

We are now ready to introduce our new representation of joint policies for QDec-POMDPs. We build on Knowledge-Based Programs (KBPs) (Fagin et al. 2003) and on their use as policies for single-agent conformant planning (Lang and Zanuttini 2012). In a nutshell, single-agent KBPs represent policies by “programs” built using sequential composition, and branching and iterating with conditions about what the agent knows. We first generalize them to allow branching on the last observation received, through the constructor \( jo \) (“just observed”). This allows us to properly generalize reactive representations such as policy trees.

Let \( M = (I, X, A, \Omega, T) \) be a QDec-POMDP model.
Definition 7 ((MA)KBP) A Knowledge-Based Program (KBP) for agent \( i \) is an expression generated by:
\[
\kappa ::= \epsilon \mid a \mid \kappa_i \kappa_j \mid \text{if } \Theta \text{ then } \kappa \text{ else } \kappa \mid \text{while } \Theta \text{ do } \kappa \text{ od}
\]
where \( \epsilon \) is the empty program, \( a \in A \) is an action, and \( \Theta \) is either an epistemic formula over \( X \) and \( I \) which is subjective for \( i \), or a Boolean combination of atoms of the form \( j_0(\omega) \) for observations \( \omega \in \Omega \). A Multi-Agent Knowledge-Based Program (MAKBP) is a vector of KBPs, one per agent \( i \in I \).

We sometimes enclose a KBP inside bold brackets ([...]) to promote readability.

The epistemic depth \( d(\kappa) \) of a KBP \( \kappa \) is defined to be the maximal epistemic depth of a condition \( \Theta \) occurring in it, and that of an MAKBP \( \kappa \) is defined to be the maximal epistemic depth of all individual KBPs \( \kappa_i \). We write \( \mathcal{K}^d \) for the class of all MAKBPs of epistemic depth at most \( d \). Finally, the size \( |\kappa| \) of a KBP is defined to be the number of symbols used for writing it, including the symbols used for writing the branching conditions, and \( |\kappa| \) is defined to be \( \sum_{i \in I} |\kappa_i| \).

Example 8 (continued) Let strike and radio be variables encoding that there is an air crew strike and that it is announced on the radio, resp., and let \( p_al \) encode that Alice is in the plane. The plan of Example 1 would be represented by the MAKBP \( \kappa = (\kappa_A, \kappa_B) \), with
\[
\begin{align*}
\kappa_A &= \{ \text{try-plane; if } K_A(\neg \text{plane}_A) \text{ then take-train; turn-radio-on; listen-radio; if } K_A(\neg K_B \text{ strike}) \text{ then to-airport } \} \\
\kappa_B &= \{ \text{turn-radio-on; listen-radio; if } K_B(\text{strike}) \text{ then to-station else to-airport } \}
\end{align*}
\]

Note that conceptually, \( j_0(\omega) \) can be seen as the atom \( K_i(o_i = \omega) \), if the dynamics of the problem is extended so that each state stores the last observation received by \( i \) in an extra variable \( o_i \). We also insist that \( j_0 \) constructs (resp. epistemic branching conditions) in the KBP of \( i \) refer to the last observation received by \( i \) itself (resp. to the knowledge of \( i \) itself, possibly about other agents’ knowledge). Otherwise the KBP would not make sense, as \( i \) would not be able to evaluate its branching conditions at execution time.

Operational Semantics

We now define an operational semantics for MAKBPs, namely, a model for the agents to execute them. Importantly, recall that we consider decentralized execution (each agent executes its KBP independently of the other agents), but centralized planning, so that at execution time the agents know the QDec-POMDP \( \Pi \) as well as the whole MAKBP \( \kappa \).

The operational semantics which we give is intended to be the straightforward one: each agent executes its own KBP faithfully, and all reason perfectly about the possible evolutions of the environment and of knowledge. However, providing a language which involves reasoning, as MAKBPs, with an operational semantics is a matter of choice, and we could as well study semantics involving approximate reasoning (e.g., bounded-depth epistemic reasoning). We leave this as a very interesting perspective for future work.

At execution time, when an agent \( i \) evaluates a branching condition such as \( [\text{if } K_i K_j] \), it needs to reason about the current knowledge of \( i \). This requires in particular to reason on what observations \( i' \) has collected so far, on what actions it has taken. However, due to partial observability, \( i \) may not know exactly what actions \( i' \) has taken so far, even if it knows what KBP \( i' \) is executing. To cope with this, our semantics includes reasoning about the program counters in the KBP of each agent. The intended meaning of \( (\Gamma, a, \kappa) \) is that, provided that all formulas in \( \Gamma \) are currently true, \( i \) is about to execute action \( a \), then it will execute the KBP \( \kappa \).

Definition 9 (program counter) Let \( i \) be an agent. A program counter for \( i \) is a triple \( c = (\Gamma, a, \kappa) \) with \( \Gamma = \{\theta(1), \ldots, \theta(\kappa)\} \), where each \( \theta(\kappa) \) is either an epistemic formula subjective for \( i \), or a Boolean combination of \( j_0(\omega) \) atoms, called the guard of \( c \), \( a \in A \) is called the first action of \( c \), and \( \kappa \) is a KBP called its continuation.

We will reason on what program counters may be “executed” after what others. We first define the set \( \text{FPC}(\kappa) \) of first program counters of a (nonempty) KBP \( \kappa \) as follows. If \( \kappa \) is of the form \( [a] \) (resp. \( [a ; \kappa'] \)), then \( \text{FPC}(\kappa) \) is the singleton \( \{\{a, a, c\}\} \) (resp. \( \{\emptyset, a, c\}\)). If \( \kappa \) is of the form \( [\text{if } \Theta \text{ then } \kappa \text{ else } \kappa' \text{ fi}] \), then \( \text{FPC}(\kappa) \) is defined to be \( \{\{\emptyset\} \cup \Gamma', a', \kappa''\} \cup \{\{\emptyset\} \cup \Gamma', a', \kappa''\} \} \cup \{\{\emptyset\} \cup \Gamma', a', \kappa''\} \} \}, and while constructs are handled similarly. Finally, we define the control-flow graph of \( \kappa \) intuitively, an edge \( c \rightarrow c' \) in \( \text{G}(\kappa) \) means that in an execution of \( \kappa \), the first action of \( c \) will be followed by that of \( c' \), provided the guard of \( c' \) is true.

Definition 10 (control-flow graph) Let \( \kappa \) be a KBP. The control-flow graph of \( \kappa \) is the directed graph \( \text{G}(\kappa) = (V, \rightarrow) \) where (1) \( V \) is the smallest set of program counters which contains \( \text{FPC}(\kappa) \), and contains \( \text{FPC}(\kappa') \) whenever it contains a program counter with continuation \( \kappa' \), and (2) \( \rightarrow \) contains an edge from \( c = (\Gamma, a, \kappa) \) to \( c' \) if and only if \( c' \in \text{FPC}(\kappa') \).

Example 11 (continued) Bob’s KBP \( \kappa_B \) in Example 8 has four program counters:
\[
\begin{align*}
c_0 &= (\emptyset, \text{turn-radio-on}, \{\text{listen-radio; if } \ldots \text{fi}\}), \\
c_1 &= (\emptyset, \text{listen-radio}, \{\ldots \text{fi}\}), \\
c_2 &= (\{K_B(\text{strike})\}, \text{to-station}, c), \\
c_3 &= (\{\neg K_B(\text{strike})\}, \text{to-airport}, c),
\end{align*}
\]
and \( \text{G}(\kappa_B) \) has these as vertices, with \( c_0 \rightarrow c_2 \), \( c_1 \rightarrow c_3 \), \( c_2 \rightarrow c_3 \), \( c_3 \rightarrow c_2 \).

For simplicity, we assume that there is a unique first program counter in \( \kappa \), written \( c^0(\kappa) \) or \( c^0 \) (that is, that \( \text{FPC}(\kappa) \) is a singleton); given an initial belief state \( B^0 \), this can always be enforced by precomputing the value in \( B^0 \) of all tests performed at the beginning of \( \kappa \).
With this in hand, we give an operational semantics to MAKBPs, by defining the Kripke structure in which the agents evaluate branching conditions. The following definition defines both the structure and, implicitly, how it is progressed at execution time. Intuitively, this structure reflects what agents know about the epistemic structure (of all agents) as well as about the state variables X and the program counters of all agents. Technically, it is made of all possible extended histories, where an extended t-history \( h^t \) is a sequence \( s^0 \epsilon^c \omega^t s^1 \epsilon^c \omega^t \ldots s^t \epsilon^c \) consisting of the successive states and joint observations together with the (joint) program counters executed; a unique extended history is associated to each history \( s^0 \epsilon^c \omega^t s^1 \epsilon^c \omega^t \ldots s^t \epsilon^c \) in the straightforward manner. Moreover, two extended histories are indistinguishable by an agent if and only if they have generated the same observations for it.\(^6\)

**Definition 12 (structure at time t)** Let \( M \) be a QDec-POMDP model, \( B^0 \) an initial belief state, and \( \kappa \) be an MAKP. The structure for \( M, B^0, \kappa \) at time \( t \), is the Kripke structure \( S^t(\kappa) = (W^t, (\sim^t)_{i \in I}, V) \) over X, defined by induction on t as follows:

1. \( W^0 = \{ s^0 \epsilon^c | s^0 \in B^0 \} \).
2. \( \forall i \in I, h^0 = W^0 \times W^0 \).
3. \( W^{t+1} \) is the set of all worlds \( h^t s^t \epsilon^c \omega^t s^{t+1} \epsilon^{t+1} \) with \( (a) h^t s^t \epsilon^c \in W^t \), \( (b) \forall i \in I, S^t(\kappa), h^t s^t \epsilon^c \models \Gamma_i \), with \( \Gamma_i \) the guard of \( c_i^t \), \( (c) s^{t+1} \epsilon^{t+1} \in T(s^t, \alpha^t) \), where \( \alpha^t \) is the joint action made of the first actions of \( c_i^t \), \( (d) \forall i \in I, c_i^t \rightarrow_{\kappa} c_i^{t+1} \).
4. \( \forall i \in I, h^t s^t \epsilon^c \omega^t s^{t+1} \epsilon^{t+1} \sim^t \sim_{t+1} h^t s^t \epsilon^c \omega^t s^{t+1} \epsilon^{t+1} \) if \( h^t s^t \epsilon^c \sim^t h^t s^t \epsilon^c \omega^t \) and \( \omega^t = \omega^{t+1} \).
5. and \( \forall \omega \in X, V(h^t s^t \epsilon^c \omega^t s^{t+1} \epsilon^{t+1}) = s^{t+1} \).

Items 1 and 2 in Definition 12 state that it is common knowledge for the agents that the initial state is one in \( B^0 \) and that they are all about to start executing their KPBPs, and that nothing more is known. In the definition of \( W^{t+1} \), item 3a states that possible worlds (extended histories) at time \( t + 1 \) are extensions of possible worlds at time \( t \); item 3b, that the program counter \( c_i^t \) was executable; item 3c, that the new state and observations are consistent with the dynamics of \( M \); and item 3d, that it is possible that \( c_i^{t+1} \) is now the current program counter of \( i \). Finally, Items 4 and 5 state that \( t \) distinguishes between histories only based on its local observations,\(^7\) and that the valuation of a world is determined by the current state of the environment.

Note that the situation at time \( t \) does not depend on the actual history, nor on the agent. It rather represents all possible evolutions of the environment and knowledge of the agents, i.e., what may have happened rather than what did happen.\(^6\)

**Example 13 (continued)** Write \( c_0^A, c_1^A \) for Alice’s program counters (\( \emptyset, \text{try-plane, if} \ldots \text{fi} \)) and \( (\overline{B}_A \sim \text{plane}) \), take-train, \ldots \), resp. The initial belief state consists of two possible worlds (we write only the relevant variables):

\[
\begin{align*}
\mathbf{h}^0 &= \text{strike}^0 (c_0^A, c_1^B) \quad \mathbf{h}^0 = \text{strike}^0 (c_0^A, c_1^B) \\
\text{At timestep 1, some possible worlds are:} \\
\mathbf{h}^1 &= \text{strike}^1 \text{plane}_A^1 \quad \text{radio}^1 (c_0^A, c_1^B) \\
\mathbf{h}^1 &= \text{strike}^1 \text{plane}_A^1 \quad \text{radio}^1 (c_0^A, c_1^B) \\
\mathbf{h}^n &= \text{strike}^1 \text{plane}_A^1 \quad \text{radio}^1 (c_0^A, c_1^B)
\end{align*}
\]

where \( h^1, h^n \) differ on whether the radio announced the strike at this step. Since the only observation is by Alice (whether plane A is true), we have \( h^1 \sim_B h^1 \sim_B h^n \) (Bob considers all three equally plausible), but \( h^1 \neq h^1 \) and \( h^1 \neq h^n \) (Alice knows whether she is in the plane). Still, we have \( h^1 \sim_B h^n \), since Alice has not listened to the radio yet. Interestingly, at this point Bob does not know whether there is a strike, but he knows that Alice does (since he knows that she observed this as a result of trying to fly).

We finally define the operational semantics of KBPs, by defining when a history is consistent with an MAKP.

**Definition 14 (operational semantics)** A t-history \( h^t \) is said to be consistent with an MAKBP \( \kappa \) if for all timesteps \( t' < t \), each agent evaluates the epistemic branching condition \( \Phi \) in its KBP by deciding \( S^t(\kappa), h^t \models \Phi \), where \( \overline{h}^t \) is the extended history associated to \( h^t \), and progresses its program counters and takes actions according to the straightforward semantics for \( i \), if \textbf{while}, and \textbf{jo} (\( \omega \)).\(^8\)

We emphasize that this semantics is well-defined. Precisely, since all epistemic conditions in the KBP of \( i \) are subjective for \( i \) (Definition 7), the test \( S^t(\kappa), \overline{h}^t \models \Phi \) is equivalent to \( i \) deciding whether the test is true at all situations which it considers actually possible, that is, to \( S^{t'}(\kappa), h^{t'} \models \Phi \) for all histories \( h^{t'} \) consistent with the observations which it has received.\(^9\) Given that each agent also has all the information needed for building \( S^t(\kappa) \) (namely \( M, B^0, \kappa \)), because planning is centralized, the test indeed makes sense for each agent individually, at execution time.

Finally, we emphasize that we do not intend to design agents which compute or embark an explicit representation of \( S^t(\kappa) \) at execution time. This notion serves only defining the operational semantics. Indeed, which action to perform at execution time can be computed from an intensional representation of \( S^t(\kappa) \), with a procedure similar to model checking for succinct dynamic epistemic logic (Charrier and Schwarzenburger 2017) (see also our Proposition 18).

Notwithstanding the fact that there is not a single “correct” semantics, let us emphasize that our operational semantics builds on the multi-agent logic of knowledge \( S_5 \).|\(^8\)

\(^6\)For the readers familiar with dynamic epistemic logic, our construction amounts to the iterated product update of the initial situation encoding common knowledge of \( B^0 \), by an event model built from the control-flow graphs of the KPBPs.

\(^7\)We do not need to specify that an agent distinguishes two histories in which it took different actions, since KPBPs are deterministic: this can occur only if it receives different observations.

\(^8\)We define condition \textbf{jo}(\( \omega \)) to evaluate to \( \bot \) at the first timestep.

\(^9\)Equivalently, the test needs only be performed in the agent’s internal, multipointed view of the situation (Aucher 2010).
and hence, that whenever agent $i$ evaluates $\Phi$, say, $K_iK_i\Phi$ to $\top$, it is indeed the case that $i'$ knows $\Phi$. Hence our semantics correctly captures faithful execution with perfect reasoning about the joint execution, by all agents.

**Expressiveness and Succinctness**

We now show that MAKBPs can represent any joint policy, at least as succinctly as standard representations like policy trees, and possibly exponentially more succinctly.

For this, we need to fix a concrete representation of QDec-POMDPs. Slightly generalizing the STRIPS-like representation proposed by Brafman, Shani, and Zilberstein (2013), we assume that for each joint action $a$, a set of quadruples $T_a = \{(\varphi^{(j)}, e^{(j)}_+, e^{(j)}_-, \omega^{(j)}) \mid j \in J\}$ is given, where each $\varphi^{(j)}$ is a propositional formula over $X$, each $e^{(j)}_+$, $e^{(j)}_-$ is a set of propositions (with $e^{(j)}_+ \cap e^{(j)}_- = \emptyset$), and each $\omega^{(j)}$ is a joint observation given as a tuple. Given a state $s$ and a joint action $a$, this encodes $T(s, a) = \{(s \oplus e^{(j)}_+ \odot e^{(j)}_-, \omega^{(j)}) \mid j \in \mathcal{I}\}$ with $s \oplus e^{(j)}_+ \odot e^{(j)}_- = s'$ s.t.

\[
\begin{cases}
\forall x \in e_+, s'[x] = \top, \\
\forall x \in e_-, s'[x] = \bot, \\
\forall x \in X \setminus (e_+ \cup e_-), s'[x] = s[x].
\end{cases}
\]

Finally, we assume that $B^0$ is given as a propositional formula over $X$, whose $B^0$ is the set of models.

Note that one can imagine other compact representations, especially representations based on influence diagrams or dynamic Bayesian networks (Boutilier, Dearden, and Goldszmidt 2000). Our results would also hold with such representations, provided they allow to efficiently check that a given history is indeed generated by the dynamics, and to efficiently progress the values of the state variables.

To compare representations, we consider a quite general definition of a $t$-policy tree for a model $\langle I, X, A, \Omega, T\rangle$ and an agent $i \in I$: an action $a \in A$ is a 0-policy tree, and if $a$ is an action, $\Omega_1, \Omega_2, \ldots, \Omega_t$ form a partition of $\Omega$, and $\tau_1, \tau_2, \ldots, \tau_k$ are $(t-1)$-policy trees, then $\tau = (a, \Omega_1 : \tau_1, \Omega_2 : \tau_2, \ldots, \Omega_k : \tau_k)$ is a $t$-policy tree. The semantics is that when an agent executes such a tree, it first takes action $a$, then upon receiving an observation $\omega$, it goes on by executing the tree $\tau_j$, where $\Omega_j$ is the subset of $\Omega$ containing $\omega$. The size $|\tau|$ (resp. $|\tau|$) of a tree $\tau$ (resp. joint tree $\tau$) is defined to be its total number of nodes.

**Proposition 15** Let $\tau$ be a joint policy tree for a QDec-POMDP model $M$. Then there is an MAKBP $\kappa$ which is equivalent to $\tau$ and has size linear in $|\tau|$.

**Proof.** For all agents $i \in I$, define $k_i$ to be $k(\tau_i)$, where $k(\tau)$ is defined inductively as follows. If $\tau$ is reduced to $a$, then $k(\tau)$ is defined to be the KBP $[a]$. Otherwise $\tau$ must be of the form $(a, \Omega_1 : \tau_1, \Omega_2 : \tau_2, \ldots, \Omega_k : \tau_k)$, and $k(\tau)$ is defined to be the KBP $[a]$ if $\bigwedge_{\omega \in \Omega} j_0(\omega)$ then $k(\tau_1)$ else if $\bigwedge_{\omega \in \Omega} j_0(\omega)$ then $k(\tau_2)$ else if $\bigwedge_{\omega \in \Omega} j_0(\omega)$ then $k(\tau_k)$ else $a$. (Note that the else clause is never entered.) Clearly, $k(\tau)$ is equivalent to $\tau$ and of size linear in $|\tau|$. □

Since by their very definition, joint policy trees can represent any policy (modulo equivalence), we have complete expressiveness as an immediate corollary of Proposition 15.

**Corollary 16** All (deterministic) joint policies for QDec-POMDPs can be represented as MAKBPs.

Importantly, the same result as Proposition 15 could be given for other reactive representations, such as finite state controllers (FSCs), were a goto construct added to the syntax of KBPs; this would indeed allow two parts of a KBP to share their continuation. Moreover, such a generalization would not change any result in this paper, and we only ignore this possibility to keep the presentation simple. Anyway, the result which we give next concerns any reactive class of representations, and hence does take FSCs into account.

**Exponential Gains**

We now exhibit a family of QDec-POMDPs for which (assuming NP $\not\subseteq$ P/poly) there is no family of valid reactive policies of polynomial size, but for which there is a poly-size family of valid MAKBPs, hence showing that the representation by MAKBPs can yield exponential gains in space.10

This result is directly inherited from the single-agent case (Lang and Zanuttini 2013, Prop. 2). However, this does not imply per se that reasoning about the other agents’ knowledge is useful as far as succinctness is concerned (reasoning about its own knowledge might have been enough). We show that this is indeed useful, and even that increasing the epistemic depth allowed for MAKBPs always properly increases succinctness. Namely, for all $d$, we show that there is a family $(\Pi_n,d)_{n \in \mathbb{N}}$ of QDec-POMDPs which has a family $(k_{n,d})_{n \in \mathbb{N}}$ of valid MAKBPs of depth $d$, but no poly-size valid MAKBPs of depth $d'$ with $d' < d$, nor any in any reactive representation (assuming NP $\not\subseteq$ P/poly).

**Construction** Broadly speaking, $\Pi_n,d$ encodes a problem in which some information passes through a series of $d$ pairs of agents $(i, i') (\ell = 0, \ldots, d-1)$, so that at each step, only one agent in the pair gets the information, as controlled by an (unobservable) variable $x_\ell$; this agent must take one of two actions, depending on the information received.

The piece of information passed is precisely $x_{\ell+1}$, that is, whether it is $i_{\ell+1}$ or $i'_{\ell+1}$ who obtained the previous information. Finally, once passed, the value of $x_{\ell+1}$ is reset, so that $i_{\ell}, i'_{\ell}$ cannot reason on its value any more. In this manner, $i_{\ell}, i'_{\ell}$ must choose their action depending on whether they know that $i_{\ell+1}$ obtained the previous information, or they know that $i'_{\ell+1}$ did; by induction, they must reason on whether they know that $i_{\ell-1}$ knows whether $i_{\ell-2}$ knows whether…, that is, on an epistemic formula of depth $d$.

Finally, for the last level, whether $i_{d-1}$ or $i'_{d-1}$ obtains the information is controlled by whether a propositional formula $\varphi$ is satisfiable, instead of a simple variable $x_{d-1}$; assuming NP $\not\subseteq$ P/poly, this prevents the last agent $i_{d}$ to have a poly-size reactive policy, as this would give a nonuniform polytime algorithm for propositional satisfiability.

More precisely, the dynamics of $\Pi_n,d$ is designed so that the sequence of actions is forced in any valid policy. For this,

10The complexity class P/poly (or “nonuniform P”) is the class of problems for which for all $n$, there is an algorithm whose description is poly-size in $n$ and which correctly decides all instances of size $n$ in polytime. The conjecture NP $\not\subseteq$ P/poly is standard.
we add a vector \( \vec{r} \) of variables to the set \( X \), meant to encode the current timestep, we add an encoding of \( s'[\vec{r}] = s[\vec{r}] + 1 \) to the effect of each joint action, and we make each joint action \( a \) lead to a sink state if taken at a disallowed timestep. All this can easily be encoded with a vector \( \vec{r} \) of logarithmic size and a polynomial number of extra quadruples \((\varphi, e, e, \ldots)\) (where \( e \) represents a dummy observation).

The rest of the construction is as follows. A set of \( n \) state variables \( x_{\varphi,0}, \ldots, x_{\varphi,n-1} \) represents propositions, and a 3CNF formula \( \varphi = \vee_{j=0}^{n-1} \neg \chi(j) \) on these variables is encoded in the state (over \( 8n^3 \times 3 \times (1 + \lceil \log n \rceil) \) Boolean variables). The next steps in any valid policy for \( \Pi_{n,d} \) force a specific agent \( i \) to modify the values of \( x_{\varphi,0}, \ldots, x_{\varphi,n-1} \) so that they encode a model \( \varphi \), if possible, and to set variable \( x_{d-1} \) to \( \top \) if and only if \( \varphi \) is satisfiable; for more details, we refer the reader to (Lang and Zanuttini 2013, Prop. 12).

The problem also has a variable \( x_{\ell} \) for \( \ell = 0, \ldots, d - 2 \), with unobservable value. At each timestep \( t \in [8n^3 + n + 1, \ell) \), both \( i \) and \( i' \) take an action which reveals the value of \( x_{\ell - 1} \) to \( i \) (resp. to \( i' \)) if \( a^H[x_{\ell}] = \top \) (resp. \( \bot \)). Moreover, except for \( x_0 \), whose value never changes, \( x_{\ell - 1} \) is reset to \( \bot \) at the same time. This transition can be encoded efficiently, through the four quadruples \((\varphi, e, e, \ldots)\) of the form \((x_{\ell - 1} \land x_{\ell}, \emptyset, \{x_{\ell - 1}, (\cdots , x_{\ell - 1}, \ldots )\}, (\forall x_{\ell - 1} \land x_{\ell}, \emptyset, \{x_{\ell - 1}, (\cdots , x_{\ell - 1}, \ldots )\}, \ldots)\), etc. (where again, \( e \) represents a dummy observation).

Finally, a last agent \( i_d \) observes the CNF \( \varphi \), and the bits encoding \( \varphi \) are reset to \( \bot \). At the last timestep \( H \), the agents must together take a joint action so that

1. one of \( i_1, i_2 \) takes action \( a \) (resp. \( a' \)) if \( s^H[x_0] = \top \) (resp. \( \bot \)), and the other takes a no-op,
2. for \( \ell = 1, \ldots, d - 1 \), one of \( i_1, i_2 \) takes action \( a \) (resp. \( a' \)) if \( x_{\ell - 1} \) (resp. \( x'_{\ell - 1} \)) takes action \( a \) or \( a' \),
3. \( i_d \) takes a (resp. \( a' \)) if \( i_{d - 1} \) (resp. \( i'_{d - 1} \)) takes \( a \) or \( a' \).

Any other joint action taken at this step sets leading to asink state. The dynamics can be encoded efficiently as there are a polynomial (in \( n \)) number of joint actions allowed at all, and overall the size of \( \Pi_{n,d} \) is polynomial in \( n \).

We are now ready to state our main result.

**Proposition 17** For all \( d \geq 2 \), the family \((\Pi_{n,d})_{n \in \mathbb{N}}\) is s.t.

1. there is a family \((\kappa_{n,d})_{n \in \mathbb{N}}\) of MAKBPs of depth \( d \), so that \( \kappa_{n,d} \) is valid for \( \Pi_{n,d} \) and is poly size (in \( n \)),
2. assuming \( \text{NP} \not\subseteq \text{P/poly} \), for any class \( \mathcal{R} \) of reactive representations, there is no family \((\pi_{n,d})_{n \in \mathbb{N}}\) so that \( \pi_{n,d} \) is valid for \( \Pi_{n,d} \) and has a poly size representation in \( \mathcal{R} \),
3. for \( d' < d \), there is no family \((\kappa_{n,d'})_{n \in \mathbb{N}}\) of MAKBPs of depth at most \( d' \), so that \( \kappa_{n,d'} \) is valid for \( \Pi_{n,d} \).

**Sketch of proof.** For Item 1, the valid MAKBPs are the sequences of actions enforced by the problem. The only exceptions are for \( i_0 \), for which we use the construction of (Lang and Zanuttini 2013, Prop. 12), and for the last action, for which it is easily seen that it is enough for all agents \( i \) to execute \([\text{if } K(KW_{u-1} \cdots (KW_u x_0)) \text{ then } a \text{ else if } K(¬(KW_{u-1} \cdots (KW_u x_0)) \text{ then } a' \text{ else no-op } f)]\), which indeed has depth \( d \).

For Item 2, assume that there is such a family. Then take any 3CNF \( \varphi \) and simulate the MAKB. If \( i_d \) plays \( a \) then \( \varphi \) is satisfiable, otherwise it is unsatisfiable. This gives a poly-time, poly-size algorithm for deciding the satisfiability of a 3CNF formula over \( n \) variables, hence a nonuniform polytime algorithm for 3SAT, a contradiction.

Finally, for Item 3, we can show that for any valid MAKB \( \kappa \), the pointed structures \( S'(\kappa) \), \( \overline{h}_a \), and \( S'_d(\kappa), \overline{h}_a' \), \( (t > 8n^3 + n + d + 1) \) are bisimilar up to depth \( d - 2 \), i.e., that the subgraphs of \( S'(\kappa) \) centered at \( h_a, h_a' \), and of radius \( d - 1 \) are isomorphic (Blackburn, Rijke, and Venema 2001), whenever \( h_a \) and \( h_a' \) have the same values for \( x_i, \ell = 0, \ldots, d - 3 \), but a different one for \( x_{d-2} \) (by induction on \( d \)). Intuitively, this captures the fact that no epistemic condition \( \Phi \) of depth less than \( K_{i_d}(KW_{u-1} \cdots (KW_u x_0)) \) can tear apart histories of the form \( h_a \) (where \( i_d \) must take action \( a \)) from histories of the form \( h_a' \). As a consequence, any MAKB allowing \( i_d \) to take the correct action at the last timestep must have depth at least \( d \). □

**Algorithmic Results**

We now investigate the complexity of the main computational problems related to MAKBPs, viz. execution and verification, with input given in compact form as in the previous section. Note that the complexity of the general planning problem of deciding whether there exists an MAKB which is valid for a given QDec-POMDP is the same as for any other representation, since MAKBPs are fully expressive (Corollary 16). Precisely, it is NEXP-complete (Brafman, Shani, and Zilberstein 2013, Corollary 1).

Recall that the execution problem asks what action \( i \) should take, given \( M, P^i, \kappa, h_i \). The membership proof in the next proposition follows the same structure as that, given by (Charrier and Schwarzentruber 2017), that model checking for succinct dynamic epistemic logic is in \( \text{PSPACE} \). This is not by coincidence, as progression for MAKBPs can be seen as \( \text{DEL} \) product updates (see Footnote 6).

**Proposition 18** The execution problem for MAKBPs is \( \text{PSPACE} \)-complete.

**Proposition 19** The problem of deciding whether a given MAKB is valid for a given QDec-POMDP at a finite horizon \( H \in \mathbb{N} \), given in unary,\(^{11}\) is \( \text{PSPACE} \)-complete.

**Sketch of proof.** For membership, by definition an MAKB is not valid for \( \Pi \) if and only if there is a history consistent with it and which does not reach the goal. Since \( \text{PSPACE} \) is closed under nondeterminism and complementation, we have the result. For hardness, we reduce the execution problem (Prop. 18) to verification by using an action which sets a variable \( x_o \) to \( \top \), and defining the goal \( g \) to be \( x_o \). Then deciding whether the MAKB is valid amounts to deciding whether this action is executed, hence the result. □

These propositions show in particular that MAKBPs can be executed and verified without explicitly computing nor maintaining the (exponential) knowledge structure \( S'(\kappa) \) at execution time. Hence the gain in succinctness with respect

\(^{11}\)This amounts to considering that the input has at least size \( H \).
to reactive policies is not only at design time, when writing the policies, but also at execution time, when embarking and reasoning with them can be done in polynomial space.

Finally, since we allowed the while construct in KBPs, we also investigate verification at an indefinite horizon. The proof uses essentially the same ideas as the proof that epistemic planning with preconditions on knowledge is undecidable (Bolander and Andersen 2011). Observe however that this result is not obvious, as the same problem is decidable in the single-agent case (Lang and Zanuttini 2012, Prop. 6).

**Proposition 20** Determining if a given MAKBP terminates in finite time in all histories, for given QDec-POMDP model $M$ and initial belief state $B^0$, is undecidable.

**Perspectives**

Our first, obvious, perspective is to generalize Multi-Agent KBPs to the context of quantitative (stochastic) Dec-POMDPs (Oliehoek and Amato 2016). To this aim, we will use epistemic logic with probabilities (Fagin and Halpern 1994) and its dynamic extension (Kooi 2003; van Eijck and Schwarzentruber 2014) instead of standard epistemic logic. Doing so will make a bridge between our work and the recent introduction of occupancy state (Dibangoye et al. 2016), which can be seen as the stochastic counterpart of our knowledge structures $S^t(\kappa)$. Another interesting perspective is to investigate approximate operational semantics for MAKBPs, for instance, semantics with reasoning at bounded modal depth, which may lead to a decidable verification problem. We also think that the expressivity of our framework allows to formalize real applications, such as Hanabi. Finally, the question of how to synthesize MAKBPs (that is, to plan) is very important; for this point, we intend to follow the direction of multi-agent epistemic regression.

**Acknowledgments** This work has been made possible by the existence of the working group MAFTEC of the French CNRS pré-GDR IA. The authors want to thank in particular Frédéric Maris, Tiago de Lima, Guillaume Aucher, and Alexandre Niveau for their contribution to the working group and to the early construction of this work.

Abdallah Saffidine is the recipient of an ARC DECRA Fellowship (DE150101351).

**References**


