On the Model-Checking of Branching-time Temporal Logic with BDI Modalities

Salvatore La Torre
Università degli Studi di Salerno
Italy
slatorre@unisa.it

Gennaro Parlato
Università degli Studi del Molise
Italy
gennaro.parlato@unisa.it

ABSTRACT

BDI logics, i.e., logics with *belief, desire* and *intention* attitudes, are one of the most widely studied formal languages for modelling rational agents. In this paper, we consider the logic $\text{CTL}^*_{\text{BDI}}$ that augments the branching-time logic CTL^* with the BDI modalities and adopt the possible-world semantics by Rao and Georgeff. We recall that in this semantics BDI relations vary over time according to a branching-time structure. We study the related model-checking question for finite-state structures, and in particular, we focus on models that are described as tuples of Kripke structures (one for each world) and where the BDI relations are captured by finite-state relations. Note that for formulas that do not contain BDI modalities this corresponds to standard CTL^* model-checking that is known to be PSPACE-complete. We show that by adding the BDI modalities the computational complexity of model-checking remains PSPACE-complete. The problem is still PSPACE-hard even if we disallow the nesting of temporal operators in the path formulas, i.e., we restrict to the temporal modalities of CTL. Finally, we give a fixed-point formulation of our algorithm for CTL_{BDI} that implements it on the top of existing symbolic fixed-point solvers.

KEYWORDS

BDI logic; temporal logic; model-checking; agents.

ACM Reference Format:
Salvatore La Torre and Gennaro Parlato. 2020. On the Model-Checking of Branching-time Temporal Logic with BDI Modalities. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

The use of rational agents for modeling real world systems has been thoroughly investigated and is now well accepted. An architecture that has emerged for the study of agent-oriented systems sees such systems as rational agents having certain mental attitudes of *belief, desire*, and *intention* (BDI *agents*). Agent beliefs can be seen as the informative component of the system state, i.e., what the system knows about the state of the environment. Agent desires can be thought of as representing the motivational state of the system, i.e., the information about the objectives to be accomplished including priorities or payoffs associated with them. Agent intentions capture the deliberative component of the system, i.e., a high-level plan coming with the agent's commitment to achieve it (intentions force the agent to pursue certain desires) [8].

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

BDI agents, and the related specification languages denoted as BDI logics, have received different formulations (see [17]). Their increasing use in the design and implementation of safety-critical applications has also motivated several approaches to the verification of such models (see [3, 5–7, 16]).

In this paper, we use the approach of Rao and Georgeff [19, 20] based on a possible worlds semantics where each possible world is not an instantaneous state but rather a transition system: possible worlds share (and are synchronized over) a branching-time structure whose time points represent the instantaneous states. Belief, desire and intentions are expressed through accessibility relations that relate the possible worlds at each time point and thus can possibly vary over time. Allowing these relations to vary over time is an important modeling feature for capturing the behavior of systems (see [11] and references therein).

System properties are expressed using extensions of CTL and CTL^* [10] with the belief, desire, and intention modal operators. We denote these logics respectively as CTL_{BDI} and $\text{CTL}^*_{\text{BDI}}$. For logic CTL_{BDI}, the model-checking question, i.e., determining whether a given model satisfies a given specification, and the satisfiability question, i.e., whether a formula admits a model where it is fulfilled, have been studied respectively in [20] and [21]. In particular, in [20] a polynomial time decision algorithm is given by restricting the worlds of the system models to have only a finite number of time points. Although the use of a finite number of time points suffices to model some realistic scenarios such as those modeled as decision trees (see [21]), in general this seems to be a serious restriction when dealing with reactive systems that typically exhibit non-terminating behaviors.

In this paper, we investigate further the model-checking question for BDI branching-time logics and study the model-checking problem for CTL_{BDI} and $\text{CTL}^*_{\text{BDI}}$ over *finite-state* structures, where we assume that models are described as tuples of Kripke structures (one for each world) and where the BDI relations are captured by finite-state automata.

We show that $\text{CTL}^*_{\text{BDI}}$ model-checking over finite-state structures is in PSPACE. Our decision algorithms build a finite graph that combines the Kripke structures representing the possible worlds along with the finite automata capturing the BDI-accessibility relations. The construction consists of the synchronous cross product of all these transition systems. Thus, we can determine the fulfillment of a given formula φ by labeling each node u of the graph with the φ sub-formulas that hold true at u. This is done similarly to the standard CTL model-checking algorithm which iteratively labels the states of a Kripke structure by considering sub-formulas with increasing number of operators (see [9]). We obtain the decision algorithm for model-checking $\text{CTL}^*_{\text{BDI}}$ by extending the CTL_{BDI}

model-checking algorithm similarly to how a decision algorithm CTL^* is obtained from that for CTL [9].

For formulas without BDI modalities, $\text{CTL}^*_{\text{BDI}}$ model-checking corresponds to standard CTL^* model-checking that is known to be PSPACE-complete. Thus, we immediately get that $\text{CTL}^*_{\text{BDI}}$ model-checking is PSPACE-hard and thus PSPACE-complete. We further prove that the problem is still PSPACE-hard even if we disallow the nesting of temporal operators in the path formulas, i.e., we restrict to the temporal modalities of CTL, which shows that also CTL_{BDI} model-checking is PSPACE-complete.

Finally, we provide a fixed-point formulation of our decision algorithm for CTL_{BDI} model-checking, which enables us to implement it on top of existing symbolic fixed-point engines. In particular, we implemented a prototype tool that uses the BDD-based model checker MUCKE [4] as back-end engine, and evaluated it on a handful of small benchmarks.

The rest of the paper is organized as follows. In Section 2, we give a motivating example. Section 3 is devoted to the formal definition of the logics, while in Section 4 we define the corresponding model-checking problems. In Section 5, we present our decision algorithms, and in Section 6 we study the computational complexity of the problem. We discuss an implementation of a prototype tool and in particular give a fixed-point formulation of our CTL_{BDI} decision algorithm in Section 7. We conclude the paper with few observations in Section 8.

Preliminary results of the research reported in this paper were given in [15].

2 EXAMPLE

In this section, we illustrate our settings with a simple example that is based on an example reported in [20].

Consider a robot that can perform two tasks: getting a beer from the refrigerator and opening the door. Both tasks require two actions. The first one requires the actions "go to the refrigerator" (gf) and "bring back a can of beer" (bb), and the second one the actions "go to the house door" (gd) and "open the door and go back" (od). In case there is no beer in the refrigerator, the robot can "go back without beer" (nb). Also, the doorbell can ring at any time and this is captured by the action rng that can occur in any state. Figure 1.a shows a corresponding transition system. We identify the time positions with the sequences of actions generated by this transition system (which clearly corresponds to a tree structure given by the unwinding of the transition system).

The only uncertainties in the environment are the presence or not of a beer can in the refrigerator and of a person at the door house. We can model these as beliefs by using two atomic propositions: *br* which stands for the robot believing that "a beer can is in the refrigerator" and *prs* which stands for the robot believing that "a person is at the door house". We thus model the four possible beliefs with corresponding worlds that are obtained from the above transition system by labeling the states according to one of the possible choices for *br* and *prs*. Accordingly we denote these worlds as $w_{br,prs}, w_{br}, w_{prs}, w_\emptyset$. Figure 1.b shows world w_{br}.

In the beginning, all the four beliefs are possible and thus all the four worlds are belief-accessible. As soon as the robot realizes that no beer is in the refrigerator (taking action nb) only the two worlds

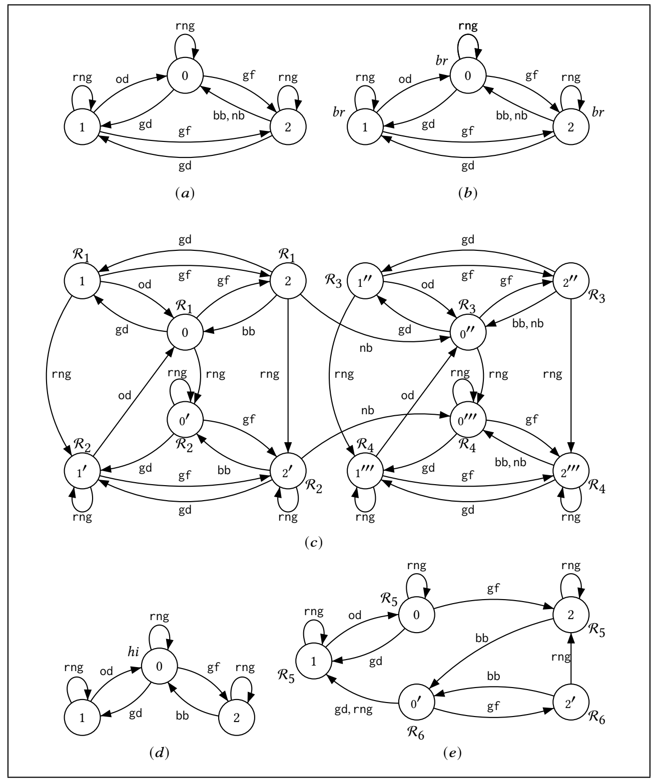

Figure 1: (a) Robot events; (b) world for the belief "there is a beer can in the refrigerator and no person at the door"; (c) belief-accessibility relation; (d) world for the desire "high reward"; (e) desire-accessibility relation.

matching this belief become accessible, i.e., worlds w_{prs} and $w_\emptyset$. Also, if the doorbell rings, the robot changes its beliefs about the presence of a person at the house door. After an action od is taken the robot becomes again agnostic on whether there is a person at the door. After a nb occurrence instead its believes about the content of the refrigerator will not change forever (this might be changed by adding a further event that a delivery man brings some beer cans). It is simple to see that the just described accessibility relation can be captured by the automaton in Figure 1.c, where denoting W the set of all worlds in our model we set $\mathcal{R}_1 = W \times W$, $\mathcal{R}_2 = W \times \{w_{br,prs}, w_{prs}\}$, $\mathcal{R}_3 = W \times \{w_{prs}, w_\emptyset\}$, $\mathcal{R}_4 = W \times \{w_{prs}\}$.

Concerning to its desires, the robot wishes both to bring a beer can back and open the door. However bringing back the beer too often gives a lower reward (a cold beer is more enjoyable when thirsty!). Also, if the robot goes to the refrigerator it does not desire to go to the door until it opens the refrigerator, and similarly the other way around. To capture this we introduce a new atomic proposition *hi* that holds true if and only if the robot gets "high reward", and capture the desires with two more worlds w_{hi} and $w_{\bar{hi}}$ that are obtained from the transition system of Figure 1.a by removing the transitions between states 2 and 1 such that we disallow the sequences containing gf.gd and gd.gf. In addition, the start state 0 of world w_{hi} is also labeled with *hi* (Figure 1.d).

The desire-accessibility relation is designed to access the worlds w_{hi} and $w_{\bar{hi}}$ depending on whether the robot is in a "high reward" or a "low reward" scenario. In Figure 1.e, we give an automaton for the desire-accessibility relation that sends to $w_{\bar{hi}}$ (by $\mathcal{R}_6 = W \times \{w_{\bar{hi}}\}$) whenever we attempt to get another beer can right after getting one, and to w_{hi} (by $\mathcal{R}_5 = W \times \{w_{hi}\}$) otherwise.

Robot intentions can be modeled by adding more worlds "refining" the desire related worlds such that for example we enforce some particular patterns of events to get a high reward. We omit a detailed discussion of this in this version of the paper.

Consider the structure $\mathcal{M}$ formed of the tree structure obtained by the unwinding of the transition system of Figure 1.a from state 0, the worlds $\{w_{br,prs}, w_{br}, w_{prs}, w_{\emptyset}, w_{hi}, w_{\bar{hi}}\}$, and the BDI accessibility relations defined by the labels $\mathcal{R}_1, \ldots, \mathcal{R}_6$. Starting from state 0 of world w_{br}, a sample of a property that is fulfilled is "whenever the robot believes that a beer can is in the refrigerator, she can possibly bring it back" that can be expressed in CTL_{BDI} as $\forall\Box(\text{ BEL } br \rightarrow \exists\Diamond \text{bb})$. A sample of a property that is not fulfilled instead is "whenever the robot believes that a beer can is in the refrigerator, she can always bring it back" that can be expressed in CTL_{BDI} as $\forall\Box(\text{ BEL } br \rightarrow \exists\Box \text{bb})$.

3 PRELIMINARIES

We briefly recall the definitions of CTL_{BDI} and $\text{CTL}^*_{\text{BDI}}$ [20]. Henceforth, for an integer $k > 0$, $[k]$ will denote the set $\{1, \ldots, k\}$.

3.1 Syntax

A $\text{CTL}^*_{\text{BDI}}$ formula can be a *state* or a *path* formula. State formulas are inductively defined starting from atomic propositions by applying the logical connectives, the path quantifiers (to path formulas) and the *belief* (BEL), *desire* (DES), and *intention* (INT) operators. Path formulas are either state formulas or obtained by applying temporal operators such as *next* ($\bigcirc$) and *until* ($\mathcal{U}$). Formally, the syntax of $\text{CTL}^*_{\text{BDI}}$ is as follows:

Definition 3.1 ($\text{CTL}^*_{\text{BDI}}$ syntax). Let AP be the set of atomic propositions. A state formula is inductively defined as follows:

- p is a state formula, for $p \in AP$;
- $\neg\varphi$ and $\varphi \vee \psi$ are state formulas, for state formulas φ and ψ;
- $\exists\varphi$ and $\forall\varphi$ are state formulas, for a path formula φ;
- $\text{BEL}\varphi$, $\text{DES}\varphi$, and $\text{INT}\varphi$ are state formulas, for a state formula φ.

Moreover, a path formula is either a state formula or any of $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$, and $\varphi\,\mathcal{U}\psi$ where φ and ψ are path formulas. A $\text{CTL}^*_{\text{BDI}}$ formula is any state formula generated by the above rules. □

The syntax of CTL_{BDI} can be obtained from that of $\text{CTL}^*_{\text{BDI}}$ by disallowing the nesting of Boolean and temporal operators in the path formulas. Namely, in CTL_{BDI} a path formula is just either $\bigcirc\varphi$ or $\varphi\,\mathcal{U}\psi$, for state formulas φ and ψ. Other operators such as $\rightarrow$, $\Box$, $\Diamond$ can be obtained as abreviations of the above ones as usual [9].

3.2 Semantics

The meaning of $\text{CTL}^*_{\text{BDI}}$, and thus of CTL_{BDI}, formulas is defined according to a possible world semantics where each possible world is not an instantaneous state but a transition system. All possible worlds share (and are synchronized over) a branching-time structure whose time points represent the instantaneous states. The meaning of the belief-desire-intention (BDI for short) operators is given through accessibility relations that relate the possible worlds at each time point and thus can possibly vary over time. The meaning of temporal operators is instead related to the (temporal) accessibility relation defined by the branching-time structure.

We now recall the notion of tree-structure. For $k > 0$, a k-ary *tree-structure* is a pair $(\mathcal{T}, \mathcal{R})$ where $\mathcal{T} \subseteq [k]^*$ is a prefix-closed set and $\mathcal{R} = \{(t, t') \mid t, t' \in \mathcal{T} \text{ and } t' = t.i \text{ for } i \in [k]\}$. Note that the empty word ε denotes the root of the tree. In the following, we will refer to the elements of $\mathcal{T}$ as *time points* and to ε as *root*. Moreover, $\mathcal{T}$ is assumed to be *infinite* unless otherwise specified.

A structure is formally defined as follows:

Definition 3.2 (Structures). A *structure* for $\text{CTL}^*_{\text{BDI}}$ (resp.,CTL_{BDI}) formulas is a tuple $\mathcal{M} = (AP, \mathcal{T}, \mathcal{R}, \mathcal{W}, \mathcal{B}, \mathcal{D}, \mathcal{I})$ where:

- AP is a set of atomic propositions;
- $(\mathcal{T}, \mathcal{R})$ is a tree-structure;
- $\mathcal{W}$ is a set of possible worlds where each world $w \in \mathcal{W}$ is a tuple $(\mathcal{T}_w, \mathcal{R}_w, \mathcal{L}_w)$ where $\mathcal{T}_w \subseteq \mathcal{T}$, $\mathcal{R}_w$ is the restriction of $\mathcal{R}$ to $\mathcal{T}_w$, $(\mathcal{T}_w, \mathcal{R}_w)$ is a tree-structure and $\mathcal{L}_w : \mathcal{T}_w \rightarrow 2^{AP}$ assigns a set of atomic propositions to each time point in w;
- for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$, $\mathcal{K} \subseteq \mathcal{W} \times \mathcal{T} \times \mathcal{W}$ is such that for $(w, t, v) \in \mathcal{K}$, $t \in \mathcal{T}_w \cap \mathcal{T}_v$ must hold (i.e., BDI accessibility relations are consistently defined with respect to the world time points).

$\mathcal{R}$ (resp., $\mathcal{B}$, $\mathcal{D}$, $\mathcal{I}$) is called the temporal (resp., belief, desire, intention) accessibility relation. □

A *path* π in a world $w = (\mathcal{T}_w, \mathcal{R}_w, \mathcal{L}_w)$ is a sequence of time points $t_0 t_1 \ldots$ such that $(t_i, t_{i+1}) \in \mathcal{R}_w$ for $i \geq 0$. The meaning of formulas is given by the satisfaction relation (see below) that is defined starting from a time point for state formulas and along a path for path formulas.

Definition 3.3 ($\text{CTL}^*_{\text{BDI}}$ semantics). For a world $w \in \mathcal{W}$ and a structure $\mathcal{M} = (AP, \mathcal{T}, \mathcal{R}, \mathcal{W}, \mathcal{B}, \mathcal{D}, \mathcal{I})$, the satisfaction relation $\models$ is inductively defined as follows (where $t \in \mathcal{T}_w$, $\pi = t_0 t_1 \ldots$ is a path of w and $\pi_i = t_i t_{i+1} \ldots$ is the suffix of π from t_i):

- $\mathcal{M}, w, t \models p$ iff $p \in \mathcal{L}_w(t)$ (where $p \in AP$);
- $\mathcal{M}, w, t \models \neg\varphi$ iff $\mathcal{M}, w, t \not\models \varphi$;
- $\mathcal{M}, w, t \models \varphi \vee \psi$ iff $\mathcal{M}, w, t \models \varphi$ or $\mathcal{M}, w, t \models \psi$;
- $\mathcal{M}, w, t \models \exists\varphi$ iff there is a path π' of w starting from t such that $\mathcal{M}, w, \pi' \models \varphi$;
- $\mathcal{M}, w, t \models \forall\varphi$ iff for all paths π' of w from t, $\mathcal{M}, w, \pi' \models \varphi$;
- $\mathcal{M}, w, t \models \text{BEL}\varphi$ iff for all $v \in \mathcal{W}$ such that $(w, t, v) \in \mathcal{B}$, it must hold $\mathcal{M}, v, t \models \varphi$ (similarly for $\text{DES}\varphi$ and $\text{INT}\varphi$);
- if φ is a state formula, $\mathcal{M}, w, \pi \models \varphi$ iff $\mathcal{M}, w, t_0 \models \varphi$;
- $\mathcal{M}, w, \pi \models \neg\varphi$ iff $\mathcal{M}, w, \pi \not\models \varphi$;
- $\mathcal{M}, w, \pi \models \varphi \vee \psi$ iff $\mathcal{M}, w, \pi \models \varphi$ or $\mathcal{M}, w, \pi \models \psi$;
- $\mathcal{M}, w, \pi \models \bigcirc\varphi$ iff $\mathcal{M}, w, \pi' \models \varphi$ where $\pi' = t_1 \ldots$;
- $\mathcal{M}, w, \pi \models \varphi\,\mathcal{U}\psi$ iff there is a $j \geq 0$ such that $\mathcal{M}, w, \pi_j \models \psi$ and $\mathcal{M}, w, \pi_i \models \varphi$ for $0 \leq i < j$.

We say $\mathcal{M}$ *satisfies* a $\text{CTL}^*_{\text{BDI}}$ formula φ at a world w, written as $\mathcal{M}, w \models \varphi$, iff $\mathcal{M}, w, root \models \varphi$. □

4 MODEL-CHECKING

We define our model-checking problem over finite-state structures where the accessibility relations are captured by finite-state transition systems. In particular, we assume a finite number of possible worlds where each world corresponds to the unrolling of a Kripke structure (a finite-state transition system whose states are labeled with atomic propositions). Furthermore, the BDI relations are captured by a finite automaton over the paths of the corresponding tree-structures.

We start recalling the definition of Kripke structures[1]. For a set of atomic propositions AP and a posistive integer k, a *Kripke structure* K of arity k is a triple (S, ν, λ) where S is a finite set of states, $\nu : S \times [k] \to S$ is a partial successor function that assigns to each state its i-successor state if any for $i \in [k]$, and $\lambda : S \to 2^{AP}$ is a labeling function.

From K and a state $s \in S$, we can define a corresponding tree-structure $\tau(K, s) = (\mathcal{T}, \mathcal{R})$ by unrolling the loops of K and taking s as the root. Formally, $\tau(K, s)$ is inductively defined as the minimal k-ary tree-structure such that (we also define a function $\tau_{K,s}$ that maps time points to corresponding states of the Kripke structure): (1) $root \in \mathcal{T}$ and $\tau_{K,s}(root) = s$, and (2) for $t \in \mathcal{T}$, if $\nu(\tau_{K,s}(t), i) = s''$ then $t.i \in \mathcal{T}$ and $\tau_{K,s}(t.i) = s''$.

We assume that the reader is familiar with the main definitions of finite automata (see [12]). For a finite automaton A and a state s, we denote with $L(A, s)$ the language accepted by A assuming s as the *sole* accepting state (i.e., the language accepted by A is $L(A) = \bigcup_{s \in F} L(A, s)$ where F is the accepting set of A).

For a set W and a tree-structure $(\mathcal{T}, \mathcal{R})$, a relation $\mathcal{K} \subseteq W \times \mathcal{T} \times W$ is *finite-state* over $\mathcal{T}$ if there is a deterministic finite automaton A with set of states Q, and a mapping $\mu : Q \to 2^{W \times W}$ such that $\mathcal{K} = \bigcup_{s \in Q} \{(w, t, w') \mid (w, w') \in \mu(s) \text{ and } t \in L(A, s) \cap \mathcal{T}\}$. If this is the case, we also say that $\mathcal{K}$ is defined by A and μ, denoted $\mathcal{K} = rel(A, \mu)$.

We introduce the notion of finite-state structure that we will use to define the model-checking problem we wish to solve.

Definition 4.1 (Finite-state structure). A structure for $\text{CTL}^*_{\text{BDI}}$ (resp., CTL_{BDI}) formulas $\mathcal{M} = (AP, \mathcal{T}, \mathcal{R}, \mathcal{W}, \mathcal{B}, \mathcal{D}, \mathcal{I})$ is *finite state* if $\mathcal{W}$ contains a finite number of possible worlds and for some integer $k > 0$ and for $w \in \mathcal{W}$, there are Kripke structures $K_w = (S_w, \nu_w, \lambda_w)$ of arity k and states $s_w \in S_w$ such that:

- $\mathcal{T} = \cup_{w \in \mathcal{W}} \mathcal{T}_w$ and $\mathcal{R} = \cup_{w \in \mathcal{W}} \mathcal{R}_w$ where $\tau(K_w, s_w) = (\mathcal{T}_w, \mathcal{R}_w)$;
- for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$, $\mathcal{K} \subseteq \mathcal{W} \times \mathcal{T} \times \mathcal{W}$ is a finite-state relation over $\mathcal{T}$.

According to the above definition, we have that a finite-state structure $\mathcal{M} = (AP, \mathcal{T}, \mathcal{R}, \mathcal{W}, \mathcal{B}, \mathcal{D}, \mathcal{I})$ has a finite representation of the form $(AP, k, \mathcal{W}, \bar{K}, A_{\mathcal{B}}, A_{\mathcal{D}}, A_{\mathcal{I}}, \mu_{\mathcal{B}}, \mu_{\mathcal{D}}, \mu_{\mathcal{I}})$ where $k > 0$ is an integer, $\bar{K} = \{(K_w, s_w) \mid w \in \mathcal{W}\}$ and $\mathcal{K} = rel(A_{\mathcal{K}}, \mu_{\mathcal{K}})$ for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$. In the following, we will denote finite-state structures through their finite representation.

We wish to study the following decision problems.

[1]We slightly deviate from the standard definition of Kripke structure by numbering the transitions leaving from a state. This has the only purpose of facilitating the translation to a corresponding tree-structure.

Definition 4.2 ($\text{CTL}^*_{\text{BDI}}$/$\text{CTL}_{\text{BDI}}$ model-checking problem). Given a finite-state structure $\mathcal{M} = (AP, k, \mathcal{W}, \bar{K}, A_{\mathcal{B}}, A_{\mathcal{D}}, A_{\mathcal{I}}, \mu_{\mathcal{B}}, \mu_{\mathcal{D}}, \mu_{\mathcal{I}})$, a world w and a $\text{CTL}^*_{\text{BDI}}$ (resp., CTL_{BDI}) formula φ, the $\text{CTL}^*_{\text{BDI}}$ (resp., CTL_{BDI}) model-checking problem asks whether $\mathcal{M}, w \models \varphi$.

The rest of the paper is mostly devoted to show the following theorem stating the complexity of the considered problems.

THEOREM 4.3. *The $\text{CTL}^*_{\text{BDI}}$ and CTL_{BDI} model-checking problems are* PSPACE*-complete.*

Moreover, CTL_{BDI} model-checking can be solved in time exponential in the number of possible worlds, and polynomial in the size of the formula and the size of the automata capturing the BDI *relations.*

$\text{CTL}^*_{\text{BDI}}$ model-checking instead can be solved with an extra exponential time in the size of the formula.

5 DECISION ALGORITHMS

Our decision algorithms constructs a finite graph that combines the Kripke structures representing the possible worlds along with the finite automata capturing the BDI-accessibility relations. Essentially the construction consists of the synchronous cross product of all these transition systems. Such graph allows us to determine the fulfillment of a given formula φ by labeling each node u of the graph with the φ sub-formulas that hold true at u. Such a labeling can be obtained by adapting the decision algorithms given for CTL and CTL* model-checking (see [10]) which iteratively label the states of a Kripke structure by considering sub-formulas with increasing number of operators.

For the rest of this section we fix the following:

- $\mathcal{W} = \{w_1, \ldots, w_n\}$, a set of $n > 0$ worlds and
- $\mathcal{M} = (AP, k, \mathcal{W}, \bar{K}, A_{\mathcal{B}}, A_{\mathcal{D}}, A_{\mathcal{I}}, \mu_{\mathcal{B}}, \mu_{\mathcal{D}}, \mu_{\mathcal{I}})$, a finite-state structure with a set of Kripke structures $\bar{K} = \{(K_w, s_w) \mid w \in \mathcal{W}\}$ where each $K_w = (S_w, \nu_w, \lambda_w)$ has arity $k > 0$.

We define a graph $G_{\mathcal{M}}$ as follows.

The *vertices* of $G_{\mathcal{M}}$ are of the form $(w_i, s_1, \ldots, s_n, q_{\mathcal{B}}, q_{\mathcal{D}}, q_{\mathcal{I}})$ where: (1) $i \in [n]$, w_i denotes the current world, (2) for $j \in [n]$, s_j either belongs to K_{w_j} and is the current state of world w_j or is a dummy state $\perp$ denoting that the current one is not a time point of world w_j, and (3) $q_{\mathcal{K}}$ is the current state of $A_{\mathcal{K}}$ for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$.

The *edges* of $G_{\mathcal{M}}$ come from the accessibility relations of $\mathcal{M}$ and are labeled consistently: edges derived from the temporal accessibility relation are labeled with the corresponding index from $[k]$ while those derived from the accessibility relation $\mathcal{K}$ with a fresh symbol $\sigma_{\mathcal{K}}$ for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$. Formally, denote $\bar{\nu}_w$ the total function obtained by completing ν_w by assigning $\perp$ whenever it is not defined, i.e., $\bar{\nu}_w(s, j) = \nu(s, j)$ if $\nu_w(s, j)$ is defined and $\bar{\nu}_w(s, j) = \perp$ otherwise (note that $\bar{\nu}_w(\perp, j) = \perp$ for each $j \in [k]$). For vertices $u = (w_i, s_1, \ldots, s_n, q_{\mathcal{B}}, q_{\mathcal{D}}, q_{\mathcal{I}})$ and $u' = (w_{i'}, s'_1, \ldots, s'_n, q'_{\mathcal{B}}, q'_{\mathcal{D}}, q'_{\mathcal{I}})$ of $G_{\mathcal{M}}$, we let (u, γ, u') be an edge of $G_{\mathcal{M}}$ if and only if either one of the following cases holds (we assume that components of u' equals the corresponding ones from u unless differently specified):

- $\gamma \in [k]$, $i' = i$, $s'_j = \bar{\nu}_{w_i}(s_j, \gamma)$ for $j \in [n]$, and $(q_{\mathcal{K}}, \gamma, q'_{\mathcal{K}})$ is a transition of $A_{\mathcal{K}}$ for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$ (we say that u' is a *γ-temporal successor* of u);
- $\gamma \subseteq \{\sigma_{\mathcal{B}}, \sigma_{\mathcal{D}}, \sigma_{\mathcal{I}}\}$, $\gamma \neq \emptyset$, and for $\sigma_{\mathcal{K}} \in \gamma$, $(w_i, w_{i'}) \in \mu_{\mathcal{K}}(q_{\mathcal{K}})$ (u' is said to be $\mathcal{K}$ successor of u for each $\sigma_{\mathcal{K}} \in \gamma$).

Note that each $G_{\mathcal{M}}$ vertex has at most $k+n$ successors. Denoting $\lambda_{\mathcal{M}}(u) = \lambda_{w_i}(s_i)$ for each vertex $u = (w_i, s_1, \ldots, s_n, q_{\mathcal{B}}, q_{\mathcal{D}}, q_{\mathcal{I}})$ of $G_{\mathcal{M}}$, we define the labeled graph $\mathcal{G}_{\mathcal{M}}$ as the graph $G_{\mathcal{M}}$ augmented with the labeling function $\lambda_{\mathcal{M}}$. We observe that $\mathcal{G}_{\mathcal{M}}$ differs from standard Kripke structures only for the distinction of the transitions into temporal and BDI ones. It is straightforward to extend the notation τ defined in Section 4 for Kripke structures to graphs $\mathcal{G}_{\mathcal{M}}$ by a $(k+n)$-ary tree-structure, and thus we omit further details on this. Again we denote $\tau(\mathcal{G}_{\mathcal{M}}, u)$ the tree-structure obtained from $\mathcal{G}_{\mathcal{M}}$ starting from u by unrolling the loops of $\mathcal{G}_{\mathcal{M}}$.

By $\tau(\mathcal{G}_{\mathcal{M}}, u)$, we can thus define the satisfiability of $\text{CTL}^*_{\text{BDI}}$ (and hence of CTL_{BDI}) formulas with respect to $\mathcal{G}_{\mathcal{M}}$ by treating a formula of the form $\mathcal{K}\varphi$, with $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$, as the corresponding temporal logic formula $\forall \bigcirc \varphi$ where the universal quantification is restricted to only the $\mathcal{K}$ successors of the current vertex. Analogously, standard path quantifiers are restricted to only the temporal successors. The formal definition can be easily obtained from Definition 3.3 and the above observations. Therefore, we omit it here, and again use $\mathcal{G}_{\mathcal{M}}, t \models \varphi$ (resp., $\mathcal{G}_{\mathcal{M}}, \pi \models \varphi$) meaning that φ holds in $\mathcal{G}_{\mathcal{M}}$ starting from time point t (resp., along path π).

From the given semantics, we have that the model-checking problem for $\text{CTL}^*_{\text{BDI}}$ (resp., CTL_{BDI}) reduces to the corresponding question on the labeled graph $\mathcal{G}_{\mathcal{M}}$. For a world w, we define the *initial* vertex of $\mathcal{G}_{\mathcal{M}}$ corresponding to w the only vertex of the form $(w, s_{w_1}, \ldots, s_{w_n}, q^0_{\mathcal{B}}, q^0_{\mathcal{D}}, q^0_{\mathcal{I}})$ where $q^0_{\mathcal{K}}$ is the initial state of $A_{\mathcal{K}}$ for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$ (recall that each s_{w_i} is the state coupled with the Kripke structure K_{w_i} in the finite-state structure we have fixed earlier in this section).

LEMMA 5.1. *For a world w and a $\text{CTL}^*_{\text{BDI}}$ formula φ, we get:*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{G}_{\mathcal{M}}, u \models \varphi,$$

where u is the initial state of $\mathcal{G}_{\mathcal{M}}$ corresponding to w.

A crucial property of $\mathcal{G}_{\mathcal{M}}$ is that as for standard Kripke structures the truth of branching-time state formulas depends only on the state, i.e., a state formula φ is true at a time point t of $\tau(\mathcal{G}_{\mathcal{M}}, u)$ if and only if it is true at any other time point t' such that $\tau_{\mathcal{G}_{\mathcal{M}},u}(t) = \tau_{\mathcal{G}_{\mathcal{M}},u}(t')$. To see this, for a k-ary tree-structure $(\mathcal{T}, \mathcal{R})$, we define the *abstract subtree* rooted at $t \in \mathcal{T}$ as the k-ary tree-structure $(\mathcal{T}', \mathcal{R}')$ where $\mathcal{T}' = \{t' \mid t.t'\}$ and $\mathcal{R}' = \{(t', t'.i) \in \mathcal{R} \mid t' \in \mathcal{T}'\}$. Thus, directly from the definition of $\tau(\mathcal{G}_{\mathcal{M}}, u)$, we get that the abstract subtrees rooted at t and t' coincide for all time points t, t' of $\tau(\mathcal{G}_{\mathcal{M}}, u)$ such that $\tau_{\mathcal{G}_{\mathcal{M}},u}(t) = \tau_{\mathcal{G}_{\mathcal{M}},u}(t')$, and hence the property stated above holds.

LEMMA 5.2. *Given a $\text{CTL}^*_{\text{BDI}}$ state formula φ, for all time points t, t' such that $\tau_{\mathcal{G}_{\mathcal{M}},u}(t) = \tau_{\mathcal{G}_{\mathcal{M}},u}(t')$ we get:*

$$\tau(\mathcal{G}_{\mathcal{M}}, u), t \models \varphi \text{ iff } \tau(\mathcal{G}_{\mathcal{M}}, u), t' \models \varphi.$$

The above lemma allows us to give for our model-checking questions two fixed-point decision algorithms in the style of those given for CTL and CTL*. Such algorithms proceed bottom-up on the syntactic structure of φ and starting from the labeling given by the truth of the atomic propositions, progressively label each vertex u of the graph with the sub-formulas that holds true there. The rules of the algorithm for CTL_{BDI}, denoted $\texttt{Alg-}\text{CTL}_{\text{BDI}}$, are given in Figure 2.

To get a decision algorithm for $\text{CTL}^*_{\text{BDI}}$ we can reason similarly to how a decision algorithm CTL* is obtained from that for CTL (see [9] for details). In particular, for a path formula φ denote with φ' the LTL formula obtained by replacing in φ its state sub-formulas with new atomic propositions. Thus, the truth of φ at a vertex u of $\mathcal{G}_{\mathcal{M}}$ is determined by a query to an LTL model-checking algorithm on φ' by taking for the added atomic proposition the evaluation given by *lab* to the corresponding state formulas. We denote with $\texttt{Alg-}\text{CTL}^*_{\text{BDI}}$ the resulting decision algorithm.

Let $\mathcal{M} = (AP, k, \mathcal{W}, \bar{K}, A_{\mathcal{B}}, A_{\mathcal{D}}, A_{\mathcal{I}}, \mu_{\mathcal{B}}, \mu_{\mathcal{D}}, \mu_{\mathcal{I}})$ where:
$\bar{K} = \{(K_w, s_w) \mid w \in \mathcal{W}\}$ and $K_w = (S_w, \nu_w, \lambda_w)$.

Initialization.
For each vertex u of $\mathcal{G}_{\mathcal{M}}$, set $lab(u) = \lambda_{\mathcal{M}}(u)$.

Update rules.
For each vertex u of $\mathcal{G}_{\mathcal{M}}$:

(1) if $\varphi = \neg\psi$ then $\varphi \in lab(u)$ iff $\psi \notin lab(u)$;
(2) if $\varphi = \varphi_1 \vee \varphi_2$ then $\varphi \in lab(u)$ iff either $\varphi_1 \in lab(u)$ or $\varphi_2 \in lab(u)$;
(3) if $\varphi = \exists \bigcirc \psi$, then $\varphi \in lab(u)$ iff there is a temporal successor u' of u such that $\psi \in lab(u')$;
(4) if $\varphi = \forall \bigcirc \psi$, then $\varphi \in lab(u)$ iff for all temporal successors u' of u it holds that $\psi \in lab(u')$;
(5) if $\varphi = \exists(\varphi_1 \mathcal{U} \varphi_2)$, then $\varphi \in lab(u)$ iff either $\varphi_2 \in lab(u)$, or $\varphi_1 \in lab(u)$ and there is a temporal successor u' of u such that $\varphi \in lab(u')$;
(6) if $\varphi = \forall(\varphi_1 \mathcal{U} \varphi_2)$, then $\varphi \in lab(u)$ iff either $\varphi_2 \in lab(u)$, or $\varphi_1 \in lab(u)$ and for all temporal successors u' of u it holds that $\varphi \in lab(u')$;
(7) if $\varphi = \text{BEL}\psi$, then $\varphi \in lab(u)$ iff for all $\mathcal{B}$ successors u' of u it holds that $\psi \in lab(u')$ (similarly for $\text{DES}\varphi$ and $\text{INT}\varphi$).

Figure 2: Fixed-point decision algorithm $\texttt{Alg-}\text{CTL}_{\text{BDI}}$ for CTL_{BDI} model-checking.

The correctness of algorithms $\texttt{Alg-}\text{CTL}_{\text{BDI}}$ and $\texttt{Alg-}\text{CTL}^*_{\text{BDI}}$ is a consequence of Lemmas 5.1 and 5.2, and the above observations. Thus we have:

LEMMA 5.3. *Given a CTL_{BDI} (resp., $\text{CTL}^*_{\text{BDI}}$) state formula φ, a finite-state structure $\mathcal{M}$ and a world w,*

$$\mathcal{M}, w \models \varphi \text{ iff } \varphi \in lab(u)$$

where lab is the labeling computed by $\texttt{Alg-}\text{CTL}_{\text{BDI}}$ (resp., $\texttt{Alg-}\text{CTL}^*_{\text{BDI}}$) and u is the initial state of $\mathcal{G}_{\mathcal{M}}$ corresponding to w.

6 COMPUTATIONAL COMPLEXITY

6.1 Upper bound

We observe that the construction of $\mathcal{G}_{\mathcal{M}}$ causes an exponential blow-up in the size of $\mathcal{M}$. In fact, the number of vertices of $\mathcal{G}_{\mathcal{M}}$ is $O(n\, \chi^n\, \eta^3)$ where χ is the maximum number of states over the n Kripke structures denoting the possible worlds of $\mathcal{M}$ and η is the maximum number of states over the finite-state automata denoting the BDI accessibility relations of $\mathcal{M}$. Moreover, for each vertex of $\mathcal{G}_{\mathcal{M}}$ there are at most $k+n$ outgoing edges where k is the arity of the Kripke structures. Thus, the overall number of $\mathcal{G}_{\mathcal{M}}$ edges is $O(kn^2\, \chi^n\, \eta^3)$. For a formula φ the number of its sub-formulas is linear in the size of ϕ (denoted $|\varphi|$). Thus, the fixed-point algorithm $\texttt{Alg-}\text{CTL}_{\text{BDI}}$ will converge in at most $O(|\varphi|\, kn^2\, \chi^n\, \eta^3)$ steps, and since each step require at most $O(n)$ time, we get the following:

Theorem 6.1. *The* $\mathrm{CTL}_{\mathrm{BDI}}$ *model-checking problem can be solved in time exponential in the number of possible worlds, and polynomial in the size of the formula and the size of the automata capturing the* BDI *relations.*

We recall that the LTL model-checking can be solved in time exponential in the size of the formula and linear in the size of the model [18]. Thus, the decision algorithm Alg-$\mathrm{CTL}^*_{\mathrm{BDI}}$ requires exponential time also in the size of the formula.

Theorem 6.2. *The* $\mathrm{CTL}^*_{\mathrm{BDI}}$ *model-checking problem can be solved in time exponential in the number of possible worlds and the size of the formula, and polynomial in the size of the automata capturing the* BDI *relations.*

In the following, we will argue that the algorithm Alg-$\mathrm{CTL}^*_{\mathrm{BDI}}$, and thus Alg-$\mathrm{CTL}_{\mathrm{BDI}}$, can be indeed implemented in polynomial space. The idea is to avoid the explicit labeling of the vertices of $\mathcal{G}_{\mathcal{M}}$ and use a polynomial-space oracle to recover the truth values of the state sub-formulas. This oracle can be obtained as follows.

As before, for a path formula φ denote with φ' the LTL formula obtained by replacing in φ the state sub-formulas with new atomic propositions. The oracle recovers the truth value of φ again by running the LTL model-checking algorithm on φ' but now whenever we need the truth value of a new atomic proposition we make a query recursively on the corresponding state formula. Since the LTL model-checking is PSPACE-complete [18], each such query can be done in polynomial space. Moreover, at each vertex we need to collect a number of truth values that is linear in the length of φ' and once we progress to the next vertex we can forget about the previously computed values, thus at any time we will use at most additional polynomial space for each oracle call. Furthermore, the number of oracle calls pending in the call stack at any time is bounded by the depth of the nesting of the BDI operators and path quantifiers, and thus it is at most linear in the length of the formula. Therefore, the overall additional space taken to determine the truth of state formulas at a vertex of $\mathcal{G}_{\mathcal{M}}$ is at most polynomial in the sizes of the model and the formula. Thus, we get:

Lemma 6.3. *The* $\mathrm{CTL}^*_{\mathrm{BDI}}$ *model-checking problem is in* PSPACE.

We recall that CTL model-checking is in PTIME and CTL^* model-checking is PSPACE-complete [9]. Thus, the $\mathrm{CTL}^*_{\mathrm{BDI}}$ model-checking problem is PSPACE-complete. In the next section, we show that indeed also the upper bound for $\mathrm{CTL}_{\mathrm{BDI}}$ model-checking cannot be improved.

6.2 Lower bound

We show a PSPACE lower bound for the $\mathrm{CTL}_{\mathrm{BDI}}$ model-checking problem. Our reduction is from the satisfiability problem of quantified Boolean formulas that is known to be PSPACE-complete [12]. For this you only need to use one of the BDI operators. The actual choice is irrelevant, and we will use the operator BEL.

To illustrate the reduction we fix a quantified Boolean formula ψ of the form $Q_1 x_1 . \ldots . Q_n x_n . \varphi$ where $Q_i \in \{\exists, \forall\}$ for $i \in [n]$ and φ is a Boolean formula over variables $x_1, \ldots, x_n$.

The crux of our reduction is to design a machinery that can account for all the possible valuations of $x_1, \ldots, x_n$. For this we use a different world for each variable x_i and then we use the BEL operator to collect a whole valuation of the variables. The main challenge here comes from the fact that the worlds are synchronized over time and thus we cannot just select independently the value of each variable but we also need to maintain this selection up to the vertices where the formula will be evaluated.

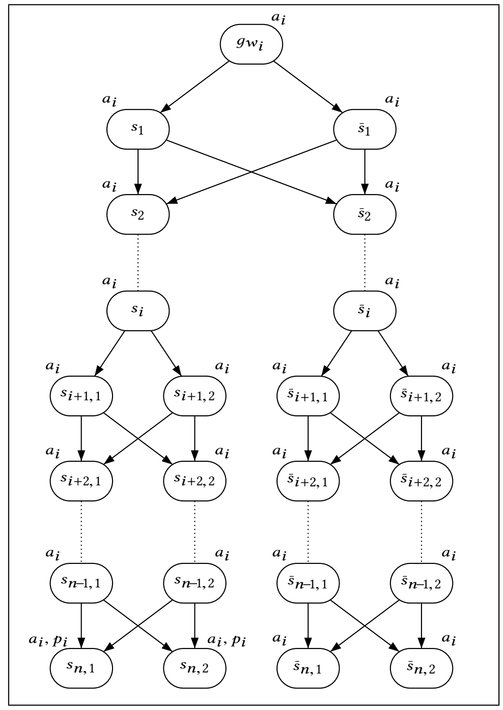

Figure 3: Graphical representation of the Kripke structure K_{w_i}.

In the following we describe in details the finite-state structure and the $\mathrm{CTL}_{\mathrm{BDI}}$ formula we construct in our reduction.

Finite-state structure. We construct a finite-state structure $\mathcal{M}_\varphi = (AP, k, \mathcal{W}, \bar{K}, A_{\mathcal{B}}, A_{\mathcal{D}}, A_{\mathcal{I}}, \mu_{\mathcal{B}}, \mu_{\mathcal{D}}, \mu_{\mathcal{I}})$ where $\mathcal{W} = \{w_1, \ldots, w_n\}$, the BDI accessibility relations do not vary over time and assign always $\mathcal{W} \times \mathcal{W}$ (all the worlds are always BDI accessible), and $\bar{K} = \{(K_{w_1}, g_{w_1}), \ldots, (K_{w_n}, g_{w_n})\}$ is described below.

Figure 3 illustrates the Kripke structures from $\bar{K}$. We use the atomic propositions $a_1, \ldots, a_n$ and $p_1, \ldots, p_n$ to label the nodes. Namely, each a_i is used to identify the nodes of K_{w_i} (thus it holds true only at the nodes of this structure) and each p_i is used to select the truth value for variable x_i. Essentially, for $i \in [n]$, K_{w_i} is a tree such that: the only leaves that can be reached from node s_i are those where p_i holds and the only ones that can be reached from node $\bar{s}_i$ are those where p_i does not hold. This can be exploited to select the truth value for variable x_i at the $i\!-\!th$ step and then maintain it till the leaves are reached. This way we can get a valuation for all the variables $x_1, \ldots, x_n$ at the time points corresponding to the leaves of $K_{w_1}, \ldots, K_{w_n}$.

Formula transformation. The starting formula ψ is transformed into a $\mathrm{CTL}_{\mathrm{BDI}}$ formula where the universal quantification over the

Boolean variables is replaced with the universal quantification over the paths, and analogously, the existential quantification over the Boolean variables with the existential quantification over the paths. Thus, the starting formula $Q_1 x_1 . \ldots . Q_n x_n . \varphi$ is transformed into a $\mathrm{CTL}_{\mathrm{BDI}}$ formula of the form $Q_1 \bigcirc \ldots Q_n \bigcirc \varphi'$ where φ' is obtained from φ by replacing each occurrence of x_i with $\mathrm{BEL}(\neg a_i \vee p_i)$.[2]

Correctness of the reduction. Denoting $\psi = Q_1 x_1 . \ldots . Q_n x_n . \varphi$ a quantified Boolean formula and $\psi' = Q_1 \bigcirc \ldots Q_n \bigcirc \varphi'$ the corresponding $\mathrm{CTL}_{\mathrm{BDI}}$ formula computed as described above, we show that ψ is valid if and only if $\mathcal{M}_\varphi, w_n \models \psi'$.

First, assume that ψ is valid. Thus, there is a valuation v of ψ that makes the formula true. On the world w_n we can replicate each choice of v for the existentially quantified variables x_i and maintain it in the corresponding world w_i as observed before. Moreover, at the leave of any possible joint path (i.e., where the worlds synchronize on the branching choices), it holds that $\mathrm{BEL}(\neg a_i \vee p_i)$ holds true if and only if we have selected p_i true at the branching corresponding to variable x_i: in fact, $\neg a_i$ holds on all worlds except for w_i and p_i holds at the reached leave of w_i if we selected the branch to s_i. Therefore, if ψ is valid, the formula ψ' holds on $\mathcal{M}_\phi$ from world w_n

For the other direction we can reason similarly, and we omit it here. Therefore, we get:

Lemma 6.4. *The $\mathrm{CTL}_{\mathrm{BDI}}$ model-checking problem is* PSPACE*-hard.*

Thus, by Lemma 6.3 and since $\mathrm{CTL}_{\mathrm{BDI}}$ is a fragment of $\mathrm{CTL}^*_{\mathrm{BDI}}$, the following theorem holds:

Theorem 6.5. *The $\mathrm{CTL}_{\mathrm{BDI}}$ and $\mathrm{CTL}^*_{\mathrm{BDI}}$ model-checking problems are* PSPACE*-complete.*

7 FIXED-POINT IMPLEMENTATION

We implemented algorithm Alg-$\mathrm{CTL}_{\mathrm{BDI}}$ in GETAFIX [13], a model-checker for (concurrent) Boolean programs that encodes the inputs and the algorithms in a fixed-point calculus and then call the fixed-point solver MUCKE [4] to evaluate it. We used the resulting prototype tool on simple benchmarks derived from the example given in Section 2. Each experiment was performed in less than a second and with negligible memory footprint.

More details will be given in a forthcoming extended version of this paper. For example, the backend fixed-point engine (mucke) uses BDD for the analysis and more care is required to determine good variable orderings.

Henceforth we illustrate the encoding of Alg-$\mathrm{CTL}_{\mathrm{BDI}}$.

7.1 Fixed-point calculus

The calculus we use is a first-order variant of the μ-calculus that has as operators Boolean combinations of sets, existential quantification over the Boolean domain, and least fixed-point operators. We start giving some notation.

A Boolean relation $R^k(x_1, \ldots, x_k)$ is any k-ary relation over the Boolean domain $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$, for some $k \in \mathbb{N}$, i.e., $R^k \subseteq \mathbb{B}^k$.

Fix a set of variables V. A Boolean expression over V is given by the following syntax:

[2] We abuse the notation and use the same symbol for both the variable quantification and the path quantification.

$$\begin{aligned} \mathit{BoolExp} \ ::= \ & T \ \mid \ F \ \mid \ R^k(x_1, \ldots, x_k) \ \mid \ \neg \mathit{BoolExp} \ \mid \\ & \mathit{BoolExp} \wedge \mathit{BoolExp} \ \mid \ \mathit{BoolExp} \vee \mathit{BoolExp} \ \mid \\ & \exists x.(\mathit{BoolExp}) \ \mid \ \forall x.(\mathit{BoolExp}) \end{aligned}$$

where $x_1, \ldots, x_k$ are variables in V, and R^k denotes any k-ary Boolean relation. The semantics of Boolean expressions is the standard one, and an expression defines some m-ary relation (where m is the number of free variables in the expression).

An equation over R is an equation of the form $R = \mathit{BoolExp}$. Note that R may appear also in *BoolExp* and thus relations may be defined recursively.

We recall that by Tarski's fixed-point theorem [22], it follows that any *positive* equation system (set of equations) has a *unique* least fixed-point (and unique greatest fixed-point). That is, there is a unique least interpretation for the relations that satisfy the equations. We assume this interpretation as the semantics for the relations defined in this calculus. More precisely, each relation R in an equation system with $R = B$ in it can be iteratively computed as follows: we start by interpreting R as the empty set, we then recursively evaluate the remaining equation system (after the substitution of R with its current interpretation) obtaining an interpretation for the other relations; we then substitute the relations contained in B with the computed interpretations thus obtaining the interpretation of R in the next iteration, and so on. The Tarski-Knaster theorem says that such iterative algorithm will always converge to the *least fixed-point* of the relations when all the expressions are positive. We observe that indeed this is the case for the expressions we use in the encoding of our algorithm Alg-$\mathrm{CTL}_{\mathrm{BDI}}$.

7.2 Algorithm encoding

From each instance of the model-checking problem, we compute a set of predicates. Namely, we have predicates: to denote the successors in the graph $\mathcal{G}_\mathcal{M}$, to relate formulas to sub-formulas, and to denote whether a sub-formula is an atomic proposition, the negation/disjunction of formulas, universally/existentially quantified, a next/until/belief/desire/intention formula. Formally, for a $\mathrm{CTL}_{\mathrm{BDI}}$ formula φ and a finite-state structure $\mathcal{M}$, we have:

(for vertices u, v, and atomic proposition p)

- `Label`(p, u) holds true iff $p \in \lambda_\mathcal{M}$;
- `Succ`$_T(u, v)$ holds true iff u is a temporal successor of v;
- for $\mathcal{K} \in \{\mathcal{B}, \mathcal{D}, \mathcal{I}\}$, `Succ`$_\mathcal{K}(u, v)$ holds true iff u is a $\mathcal{K}$-successor of v;

(for each state sub-formula ψ of φ, where $Q \in \{\forall, \exists\}$)

- `Atomic`(ψ) holds true iff ψ is an atomic proposition;
- `Neg`(ψ) holds true iff ψ is of the form $\neg\psi'$;
- `Or`(ψ) holds true iff ψ is of the form $\psi' \vee \psi''$;
- `Universal`(ψ) (resp., `Existential`(ψ)) holds true iff ψ is of the form $\forall\psi'$ (resp., $\exists\psi'$);
- `Next`(ψ) holds true iff ψ is of the form $Q \bigcirc \psi'$;
- `Until`(ψ) (resp., `Sub`(ψ', ψ'', ψ)) holds true iff ψ is of the form $Q(\psi' \mathcal{U} \psi'')$;
- `Bel`(ψ) holds true iff ψ is of the form $\mathcal{B}\psi'$; analogously for the predicates `Des` (desires) and `Int` (intentions);
- `Sub`(ψ', ψ) holds true iff ψ is of either one of the forms $\neg\psi'$, $Q \bigcirc \psi'$, $\mathcal{B}\psi'$, $\mathcal{D}\psi'$, $\mathcal{I}\psi'$, $\psi' \vee \psi''$, or $\psi'' \vee \psi'$ (i.e., ψ' is a direct state sub-formula of ψ);
- `Sub`(ψ', ψ'', ψ) holds true iff ψ is of the form $Q(\psi' \mathcal{U} \psi'')$.

$\Lambda(\sigma, \psi, u) := \Lambda^{\sigma}_{at} \vee \Lambda^{\sigma}_{\neg} \vee \Lambda^{\sigma}_{\vee} \vee \Lambda^{\sigma}_{\forall\bigcirc} \vee \Lambda^{\sigma}_{\exists\bigcirc} \vee \Lambda^{\sigma}_{\forall\mathcal{U}} \vee \Lambda^{\sigma}_{\exists\mathcal{U}} \vee \Lambda^{\sigma}_{\mathcal{B}} \vee \Lambda^{\sigma}_{\mathcal{D}} \vee \Lambda^{\sigma}_{\mathcal{I}}$ where:

(1) $\Lambda^{0}_{at} := \texttt{Atomic}(\psi) \wedge \neg\texttt{Label}(\psi, u)$

(2) $\Lambda^{1}_{at} := \texttt{Atomic}(\psi) \wedge \texttt{Label}(\psi, u)$

(3) $\Lambda^{\sigma}_{\neg} := \texttt{Neg}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi', u) \wedge \texttt{Sub}(\psi', \psi) \wedge \Lambda(1-\sigma, \psi', u))$

(4) $\Lambda^{\sigma}_{\vee} := \texttt{Or}(\psi) \wedge \exists\psi', \psi''.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \texttt{Ready}(\psi'') \wedge \texttt{Sub}(\psi'', \psi) \wedge (\Lambda(\sigma, \psi', u)\, o_{\sigma}\, \Lambda(\sigma, \psi'', u)))$, where o_{σ} is $\vee$ if $\sigma = 1$ and $\wedge$ otherwise

(5) $\Lambda^{0}_{\exists\bigcirc} := \texttt{Existential}(\psi) \wedge \texttt{Next}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \forall v.(\neg\texttt{Succ}_{T}(v, u) \vee \Lambda(0, \psi', v)))$

(6) $\Lambda^{1}_{\exists\bigcirc} := \texttt{Existential}(\psi) \wedge \texttt{Next}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \exists v.(\texttt{Succ}_{T}(v, u) \wedge \Lambda(1, \psi', v)))$

(7) $\Lambda^{0}_{\forall\bigcirc} := \texttt{Universal}(\psi) \wedge \texttt{Next}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \exists v.(\texttt{Succ}_{T}(v, u) \wedge \Lambda(0, \psi', v)))$

(8) $\Lambda^{1}_{\forall\bigcirc} := \texttt{Universal}(\psi) \wedge \texttt{Next}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \forall v.(\neg\texttt{Succ}_{T}(v, u) \vee \Lambda(1, \psi', v)))$

(9) $\Lambda^{0}_{\exists\mathcal{U}} := \texttt{Existential}(\psi) \wedge \texttt{Until}(\psi)$
$\wedge \exists\psi', \psi''.(\texttt{Ready}(\psi') \wedge \texttt{Ready}(\psi'') \wedge \texttt{Sub}(\psi', \psi'', \psi) \wedge \Lambda(0, \psi'', u) \wedge (\Lambda(0, \psi', u) \vee \forall v.(\neg\texttt{Succ}_{T}(v, u) \vee \Lambda(0, \psi, v))))$

(10) $\Lambda^{1}_{\exists\mathcal{U}} := \texttt{Existential}(\psi) \wedge \texttt{Until}(\psi)$
$\wedge \exists\psi', \psi''.(\texttt{Ready}(\psi') \wedge \texttt{Ready}(\psi'') \wedge \texttt{Sub}(\psi', \psi'', \psi) \wedge (\Lambda(1, \psi'', u) \vee (\Lambda(1, \psi', u) \wedge \exists v.(\texttt{Succ}_{T}(v, u) \wedge \Lambda(1, \psi, v)))))$

(11) $\Lambda^{0}_{\forall\mathcal{U}} := \texttt{Universal}(\psi) \wedge \texttt{Until}(\psi)$
$\wedge \exists\psi', \psi''.(\texttt{Ready}(\psi') \wedge \texttt{Ready}(\psi'') \wedge \texttt{Sub}(\psi', \psi'', \psi) \wedge \Lambda(0, \psi'', u) \wedge (\Lambda(0, \psi', u) \vee \exists v.(\texttt{Succ}_{T}(v, u) \wedge \Lambda(0, \psi, v))))$

(12) $\Lambda^{1}_{\forall\mathcal{U}} := \texttt{Universal}(\psi) \wedge \texttt{Until}(\psi)$
$\wedge \exists\psi', \psi''.(\texttt{Ready}(\psi') \wedge \texttt{Ready}(\psi'') \wedge \texttt{Sub}(\psi', \psi'', \psi) \wedge (\Lambda(1, \psi'', u) \vee (\Lambda(1, \psi', u) \wedge \forall v.(\texttt{Succ}_{T}(v, u) \wedge \Lambda(1, \psi, v)))))$

(13) $\Lambda^{0}_{\mathcal{B}} := \textsc{bel}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \exists v.(\texttt{Succ}_{\mathcal{B}}(v, u) \wedge \Lambda(0, \psi', v)))$ (similarly for desire and intention formulas)

(14) $\Lambda^{1}_{\mathcal{B}} := \textsc{bel}(\psi) \wedge \exists\psi'.(\texttt{Ready}(\psi') \wedge \texttt{Sub}(\psi', \psi) \wedge \forall v.(\neg\texttt{Succ}_{\mathcal{B}}(v, u) \vee \Lambda(1, \psi', v)))$ (similarly for desire and intention formulas)

Figure 4: Formal definition of the relation Λ.

The above predicates can be automatically computed from a formula and Boolean program representing a finite-state structure.

We capture the labeling of $\mathcal{G}_{\mathcal{M}}$ by a relation Λ given by tuples of the form (σ, ψ, u) where $\sigma \in \{0, 1\}$ denotes that: the vertex u is labeled by formula ψ, if $\sigma = 1$, and by formula $\neg\psi$, otherwise. The formal definition of Λ is given in Figure 4 where we denote as $\texttt{Ready}(\psi, u)$ the formula $\exists\sigma.\Lambda(\sigma, \psi, u)$.

Among the defined predicates, Λ is the only recursive one and is structured in disjuncts according to the type of the considered sub-formula.

The recursive evaluation of Λ will start by adding the tuples (σ, ψ, u) where ψ is an atomic proposition (Λ^{σ}_{at} is the only disjunct that does not contain Λ). According to cases (1) and (2) of Figure 4, for a vertex u, this will add all the tuples $(1, \psi, u)$ such that ψ labels u in $\mathcal{G}_{\mathcal{M}}$ and $(0, \psi, u)$ such that ψ does not. Note that after this iteration $\texttt{Ready}(\psi)$ will hold true for all atomic propositions and false for the other formulas.

In the following iterations, the other disjuncts will contribute to add tuples over formula ψ as soon as the $\texttt{Ready}$ predicate will become true for the immediate sub-formulas of ψ. In fact, such disjuncts are defined as the conjunction of two main parts: a first part that checks the type of the formula, and a second part that checks the semantics of the formula by its sub-formulas; in this second part, a main conjunct checks that the labeling of the immediate sub-formulas have been already computed (by the $\texttt{Ready}$ predicate) and the remain part ensures the semantics of ψ by the findings about its sub-formulas.

The above reasoning can be formalized in a proof by induction of the following result:

Lemma 7.1. *For each sub-formula ψ of φ and each node u of $\mathcal{G}_{\mathcal{M}}$:*

- *$\Lambda(1, \psi, u)$ holds true if and only if $\mathcal{G}_{\mathcal{M}}, u \models \psi$;*
- *$\Lambda(0, \psi, u)$ holds true if and only if $\mathcal{G}_{\mathcal{M}}, u \models \neg\psi$.*

Thus, we have the following theorem:

Theorem 7.2. *Given a $\textsc{Ctl}_{BDI}$ formula φ, a finite-state structure $\mathcal{M}$ and a world w,*

$$\mathcal{M}, w \models \varphi \text{ iff } \Lambda(1, \varphi, u) \text{ holds true}$$

where u is the initial state of $\mathcal{G}_{\mathcal{M}}$ corresponding to w.

8 CONCLUSIONS

In this paper, we have introduced a notion of finite-state structure in the possible worlds semantics by Rao and Georgeff [19, 20] and studied the related model-checking problem against $\textsc{Ctl}_{\textsc{bdi}}$ and $\textsc{Ctl}^{*}_{\textsc{bdi}}$ formulas. We have shown that these decision problems are both Pspace-complete, and have implemented and evaluated on a few benchmarks the $\textsc{Ctl}_{\textsc{bdi}}$ decision algorithm in Getafix [13]. Our results extend the decidability of the considered bdi logics to systems that exhibit infinitely many time points from the unrolling of the finite-state models.

As future research, we plan to investigate further the applications by running more experiments and implementing also our decision algorithm for $\textsc{Ctl}^{*}_{\textsc{bdi}}$. Concerning to this second aspect, we observe that an implementation can be obtained by embedding the Büchi automata for the Ltl formulas in the enconding for $\textsc{Ctl}_{\textsc{bdi}}$. We think that a more direct formulation may lead to better performances. Further, we wish to extend our results to multi-agents and modular systems where each world is composed of modules that can call each other possibly recursively, similarly to what is done for standard temporal logics (see [1, 2, 14]). This will give a more faithful representation for many real systems and will yield more succinct models (modules can be shared among worlds).

ACKNOWLEDGEMENTS

This work was partially supported by GNCS 2019-2020 and FARB-UNISA 2017-2018 grants.

REFERENCES

[1] Rajeev Alur, Michael Benedikt, Kousha Etessami, Patrice Godefroid, Thomas W. Reps, and Mihalis Yannakakis. 2005. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.* 27, 4 (2005), 786–818. https://doi.org/10.1145/1075382.1075387

[2] Benjamin Aminof, Orna Kupferman, and Aniello Murano. 2012. Improved model checking of hierarchical systems. *Inf. Comput.* 210 (2012), 68–86. https://doi.org/10.1016/j.ic.2011.10.008

[3] Massimo Benerecetti, Fausto Giunchiglia, and Luciano Serafini. 1998. Model Checking Multiagent Systems. *J. Log. Comput.* 8, 3 (1998), 401–423. https://doi.org/10.1093/logcom/8.3.401

[4] Armin Biere. 1997. μcke - Efficient μ-Calculus Model Checking. In *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings (Lecture Notes in Computer Science)*, Orna Grumberg (Ed.), Vol. 1254. Springer, 468–471. https://doi.org/10.1007/3-540-63166-6_50

[5] Jeff Blee, David Billington, Guido Governatori, and Abdul Sattar. 2011. Levels of modality for BDI Logic. *J. Applied Logic* 9, 4 (2011), 250–273. https://doi.org/10.1016/j.jal.2011.08.002

[6] Rafael H. Bordini, Louise A. Dennis, Berndt Farwer, and Michael Fisher. 2008. Automated Verification of Multi-Agent Programs. In *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy.* IEEE Computer Society, 69–78. https://doi.org/10.1109/ASE.2008.17

[7] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael J. Wooldridge. 2006. Verifying Multi-agent Programs by Model Checking. *Autonomous Agents and Multi-Agent Systems* 12, 2 (2006), 239–256. https://doi.org/10.1007/s10458-006-5955-7

[8] Michael E. Bratman. 1987. Intentions, Plans, and Practical Reason. *Harvard University Press* (1987).

[9] E. Allen Emerson. 1990. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B).* The MIT Press/Elsevier, 995–1072.

[10] E. Allen Emerson and Edmund M. Clarke. 1982. Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons. *Sci. Comput. Program.* 2, 3 (1982), 241–266.

[11] Joseph Y. Halpern and Moshe Y. Vardi. 1989. The Complexity of Reasoning about Knowledge and Time. I. Lower Bounds. *J. Comput. Syst. Sci.* 38, 1 (1989), 195–237. https://doi.org/10.1016/0022-0000(89)90039-1

[12] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. *Introduction to automata theory, languages, and computation - (2. ed.).* Addison-Wesley-Longman.

[13] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. 2009. Analyzing recursive programs using a fixed-point calculus. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2009, Dublin, Ireland, June 15-21, 2009*, Michael Hind and Amer Diwan (Eds.). ACM, 211–222. https://doi.org/10.1145/1542476.1542500

[14] Salvatore La Torre, Margherita Napoli, Mimmo Parente, and Gennaro Parlato. 2008. Verification of scope-dependent hierarchical state machines. *Inf. Comput.* 206, 9-10 (2008), 1161–1177. https://doi.org/10.1016/j.ic.2008.03.017

[15] Salvatore La Torre and Gennaro Parlato. 2019. Model Checking BDI Logics over Finite-state Worlds. In *Proceedings of the 1st Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis co-located with the 18th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2019), Rende (CS), Italy, November 19-20, 2019.* 11–16. http://ceur-ws.org/Vol-2509/paper1.pdf

[16] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT* 19, 1 (2017), 9–30. https://doi.org/10.1007/s10009-015-0378-x

[17] John-Jules Ch. Meyer, Jan Broersen, and Andreas Herzig. 2015. BDI Logics. In *Handbook of Epistemic Logic.* College Publications, 453–498.

[18] Amir Pnueli. 1977. The Temporal Logic of Programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977.* 46–57. https://doi.org/10.1109/SFCS.1977.32

[19] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991.*, James F. Allen, Richard Fikes, and Erik Sandewall (Eds.). Morgan Kaufmann, 473–484.

[20] Anand S. Rao and Michael P. Georgeff. 1993. A Model-Theoretic Approach to the Verification of Situated Reasoning Systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, Ruzena Bajcsy (Ed.). Morgan Kaufmann, 318–324. http://ijcai.org/Proceedings/93-1/Papers/045.pdf

[21] Anand S. Rao and Michael P. Georgeff. 1998. Decision Procedures for BDI Logics. *J. Log. Comput.* 8, 3 (1998), 293–342. https://doi.org/10.1093/logcom/8.3.293

[22] Alfred Tarski. 1955. A lattice-theoretical fixpoint theorem and its application. *Pacific J. Math.* 5, 2 (1955), 285–309. https://doi.org/10.2140/pjm.1955.5.285