Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 126 (2005) 93–114
www.elsevier.com/locate/entcs

Bounded Model Checking for Deontic Interpreted Systems

Bożena Woźna [a,1,2] Alessio Lomuscio[a,1,3] Wojciech Penczek[b,4]

[a] *Department of Computer Science*
University College London
Gower Street, London WC1E 6BT, United Kingdom

[b] *Institute of Computer Science, PAS*
Ordona 21, 01-237 Warsaw, Poland

Abstract

We propose a framework for the verification of multi-agent systems' specification by symbolic model checking. The language CTLKD (an extension of CTL) allows for the representation of the temporal evolution of epistemic states of the agents, as well as their correct and incorrect functioning behaviour. We ground our analysis on the semantics of deontic interpreted systems. The verification approach is based on an adaption of the technique of bounded model checking, a mainstream approach in verification of reactive systems. We test our results on a typical communication scenario: the bit transmission problem with faults.

Keywords: Bounded model checking, epistemic logic.

1 Introduction

The task of software engineers is to design and deploy a computer system that meets a particular set of specifications. This is by no means a trivial task. Only in the past few months the problems with NASA's and ESA's Mars exploration missions have made news showing how many unforeseen problems

[1] The authors acknowledge support from EPSRC (grant GR/S49353), and the Nuffield Foundation (grant NAL/690/G).
[2] Email: bozena@dcs.kcl.ac.uk
[3] Email: A.Lomuscio@ucl.ac.uk
[4] Email: penczek@ipipan.waw.pl

doi:10.1016/j.entcs.2004.11.015

even a thoroughly-designed distributed system may encounter. In mission-critical software as well as in particularly sensitive applications such as Internet protocols, the worldwide electronic banking system, etc., software engineers are interested in analysing the properties of software, and in particular in checking whether particular conditions, from the basic ones of deadlock to more complex ones, hold for a particular system.

The area of multi-agent systems is also interested in a similar set of problems. Multi-agent systems [17] are distributed systems in which the individual processes, or *agents*, are autonomous entities that engage in social activities such as coordination, negotiation, cooperation, etc. Since multi-agent systems are autonomous and social, their range of possible behaviours is even greater than the one of traditional distributed systems. It follows from this that the issue of verification of the properties a system satisfies is just as important in multi-agent systems.

Software validation, i.e., the process of checking that a piece of software satisfies certain characteristics, is currently conducted by means of three main techniques: testing, theorem proving, and model checking. Testing involves searching the state-space of the possible inputs of a program looking for potentially problematic outputs. Theorem proving techniques are based on the representation of a program by means of a system of formal logic; in its simplest instance, checking whether a property is satisfied amounts to checking whether a formula is a theorem of the logic that represents the program. Model checking in its mainstream approach involves representing all possible computational traces of a program by means of a temporal model (appropriately represented) and checking whether or not a temporal formula, representing the property to be verified, holds in this model.

Software validation has in other words an intrinsic "deontic connotation". It amounts to checking whether the system under consideration behaves as it is prescribed by its specification. But it should be noted that, in this approach, this is a property that is *external* from the logical system. One could say, the correct functioning behaviour of the system is a *metalogical* property of the logic system representing the program. Differently from what happens in deontic logic, deontic concepts here are not explicitly used in the logic to represent the system, but they are built into the procedures that operate on the logic used to check the system.

Although a range of systems have been verified by means of standard verification techniques, multi-agent systems applications call for a refined approach. Consider for example a number of automatic agents bidding for goods on an electronic auction. There may be rules as to how these agents may conduct their bidding, but it is often unfeasible and/or counterproductive to have

many of these rules hard-wired in the auction protocol itself. As designers of the auction we may consider it beneficial that agents do not bid several times a second on the same good not to have resulting denial-of-service problems at server-level, but it seems difficult to enforce this when the agents are programmed by a variety of software houses on which we have no control. Other examples from traditional federated databases to more recent fault-control modules in mission-critical software point to similar conclusions: *it is important to reason about the properties that hold in a system when the programs are functioning following their specification but also (and occasionally even more importantly) even they do not.* In other words, not only we would like to check whether a system satisfies its specification, but also we would like to derive the consequences resulting from the system not behaving as intended. Different deontic logics have been used to bring to the logical object level the distinction between correct (or ideal, normative, etc.) and incorrect states. In this paper we would like to take these ideas one step further and provide a technique by means of which we can not not specify but also automatically verify properties expressing compliance of a multi-agent system with respect to specifications.

To carry out this analysis we use the formal machinery of verification by model checking [4], and in particular the one of *Bounded Model Checking* via SAT translation [3]. In verification by model checking one typically describes a system S by means of a program in a language such as SMV [12]. This description is then supplied to the model checker which produces the (appropriately encoded) temporal model M_S representing all the possible executions of system S. To check whether a property P is satisfied in S one checks whether the temporal model M_S satisfies a formula ϕ_P representing P, i.e., $M_S \models \phi_P$. The key problem in this approach is to manage the representation of the resulting model M_S. One of the techniques available to keep the approach feasible is bounded model checking. This technique focuses on the attempt to discover faults in the specification of the system. Rather than checking the whole state space for the verification of the property, in bounded model checking one checks whether the negation of the property is actually satisfied on a fraction of the model, thereby producing a counterexample. Furthermore, the actual check is translated into a standard propositional satisfiability problem by computing appropriate translations into propositional formulas representing both the truncated model, and the formula to be checked. By means of this approach subtle bugs in protocols for reactive systems have been discovered. We refer to [1,3,5,14] for more details.

While in reactive systems it is enough to model a system by means of a purely temporal language, multi-agent systems are defined following what is

often referred to as the "intentional stance" [6]. In other words it is useful to describe autonomous agents in terms of their knowledge, belief, intentions, social context, etc. This implies that the model checking problem for multi-agent systems cannot simply be stated as one on temporal logic but that richer formalisms need being used. In past research we have provided a model checking algorithm for a branching time temporal-epistemic logic (CTLK) [13]. In this paper we extend this work by providing a bounded model checking algorithm for a logic that comprises knowledge and a deontic component representing correct functioning behaviour of the system.

The paper is organised as follows. In Section 2, we fix the notation on the semantics of deontic interpreted systems. In Section 3 we present the language of CTLKD, an extension of CTLK, representing correct/incorrect functioning behaviour of the agents. In Section 4 we present a bounded semantics definition for satisfaction that we use in Section 5 to define the algorithm of bounded model checking. In Section 6 we apply the formalism to an example close to the multi-agent systems literature: the bit transmission problem with faults.

2 Deontic Interpreted Systems

In this section we introduce *deontic interpreted systems*. These were defined in [10] to represent and reason about correct functioning behaviour of multi-agent systems. They provide a semantics based on the computation states of the agents, on which it is possible to interpret a modality $\mathcal{O}_i\phi$, representing the fact "in all correct functioning executions of agent i, ϕ holds", as well as a traditional epistemic modality $K_i\phi$ representing knowledge of ϕ by agent i, and standard branching time temporal operators[5]. An axiomatisation of deontic interpreted systems has been provided for the non-temporal fragment of the language; we refer the interested reader to [11] for more details.

The following is reported to fix the notation only; more details can be found in [10,13]. Let $\mathcal{PV}$ be a set of propositional variables and $\mathcal{A} = \{1, \ldots, n\}$ be a set of agents. Consider n non-empty sets $L_1, \ldots, L_n$ of local states, one for each agent of the system, and a set L_e of local states for the environment. For each agent $i \in \mathcal{A}$, consider a set of possible actions Act_i, and n protocols $P_i : L_i \to 2^{Act_i}$ representing the functioning behaviour of every agent, and $P_e : L_e \to 2^{Act_e}$ for the environment.

Further assume that for every agent, its set of local states can be partitioned into allowed and disallowed states. We call these states *green* and *red* respectively. For n agents and $n + 1$ mutually disjoint and non-empty sets

[5] Temporal operators were not actually used in [10] but this is a straightforward extension.

$\mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{G}_e$ we define the set S of all possible *global states* as the Cartesian product $L_1 \times \ldots \times L_n \times L_e$, such that $L_1 \supseteq \mathcal{G}_1, \ldots, L_n \supseteq \mathcal{G}_n, L_e \supseteq \mathcal{G}_e$. $\mathcal{G}_e$ is called the set of green states for the environment, and for any agent i, $\mathcal{G}_i$ is called the set of green states for agent i. The complement of $\mathcal{G}_e$ with respect to L_e (denoted by $\mathcal{R}_e$) is called the set of red states for the environment, and similarly the complement $\mathcal{G}_i$ with respect to L_i (denoted by $\mathcal{R}_i$) is called the set of red states for the agent i.

We can model the computation taking place in the system by means of a transition function $t : S \times Act \to S$, where $Act \subseteq Act_1 \times \ldots \times Act_n \times Act_e$ is the set of joint actions. Intuitively, given an initial state $\iota \in S$, the sets of protocols, and the transition function, we can build a (possibly infinite) structure that represents all the possible computations of the system. Indeed, we will deal with systems, in which the state space consists of reachable global states only. A state $s \in S$ is *reachable* if there is a sequence of states $(s_0, \ldots, s_n)$ such that $s_0, \ldots, s_n \in S$, $s_0 = \iota$, $s_n = s$, and for all $i \in \{0, \ldots, n-1\}$ there exists an action $act_i \in Act$ such that $t(s_i, act_i) = s_{i+1}$, i.e., s_{i+1} is the result of applying the transition function t to the global state s_i, and a joint action act_i. If each of the components of act_i is prescribed by the corresponding protocol P_j at agent j's local state at s_i, then the resulting state will only contain green local states, otherwise it may contain some red local states. For further considerations on this see [10]. In the following we abstract from the transition function, the actions, and the protocols, and simply use the relation T, but it should be clear that this is uniquely determined by the interpreted system under consideration.

Let $l_i : S \to L_i$ be a function which returns the local state of agent i from a global state. A *deontic interperted system* is defined as follows:

Definition 2.1 [Deontic Interperted System]
Given a set of agents $\mathcal{A} = \{1, \ldots, n\}$, a set of global states S, protocols, and a transition function, a *deontic interperted system (or simply a model)* is a tuple $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ where

- $\mathcal{DS} \subseteq S$ is a finite set of reachable global states for $\mathcal{A}$,
- $\iota \in \mathcal{DS}$ is an initial state,
- $T \subseteq \mathcal{DS} \times \mathcal{DS}$ is a serial[6] binary relation on $\mathcal{DS}$,
- $R_i^O \subseteq \mathcal{DS} \times \mathcal{DS}$ is a relation for each agent $i \in \mathcal{A}$ defined by: $sR_i^O s'$ iff $l_i(s') \in \mathcal{G}_i$[7],

[6] A relation $R \subseteq X \times X$ is *serial* if for all $x \in X$ there exists $y \in X$ such that xRy.
[7] Since each R_i^O only depends on the target state, for what pertains this component we could have equally defined a model by means of green local states for agent i.

- $R_i^K \subseteq \mathcal{DS} \times \mathcal{DS}$ is a relation for each agent $i \in \mathcal{A}$ defined by: $sR_i^K s'$ iff $l_i(s') = l_i(s)$,
- $\mathcal{V} : \mathcal{DS} \longrightarrow 2^{\mathcal{PV}}$ is a valuation function such that $true \in \mathcal{V}(s)$ for all $s \in \mathcal{DS}$. $\mathcal{V}$ assigns to each state a set of proposition variables that are assumed to be true at that state.

By $|M|$ we denote the number of states of M, whereas $\mathbb{N} = \{0, 1, 2, \ldots\}$ indicates the set of natural numbers, and $\mathbb{N}_+ = \{1, 2, \ldots\}$ the set of positive natural numbers.

A *computation* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $(s_i, s_{i+1}) \in T$ for each $i \in \mathbb{N}$. For a computation $\pi = (s_0, s_1, \ldots)$, let $\pi(k) = s_k$, and $\pi_k = (s_0, \ldots, s_k)$, for each $k \in \mathbb{N}$. In the rest of the paper we shall call π_k a *k-computation*. Moreover, a $k-$computation π_k is a *(k,l)-loop* if $(\pi_k(k), \pi_k(l)) \in T$ for some $0 \leq l \leq k$. We call π_k simply a *loop* if there is an $l \in \mathbb{N}$ with $l \leq k$ for which π_k is a (k,l)-loop. By $\Pi(s)$ we denote the set of all the infinite computations starting at state s, whereas by $\Pi_k(s)$ the set of all the $k-$computations starting at s.

3 The logic CTLKD

Here we fix syntax and semantics for CTLKD, an extension of CTL [7], introduced by Emerson and Clarke, enriched with modal operators representing correct functioning behaviour [11], and standard epistemic operators [8]. The bounded model checking problem of the temporal epistemic fragment of the language was analysed in [13].

Definition 3.1 [Syntax of CTLKD]
Let $\mathcal{PV}$ be a set of propositional variables also containing the symbol *true*. The set of CTLKD formulas $\mathcal{FORM}$ is defined inductively as follows:

- every member p of $\mathcal{PV}$ is a formula,
- if α and β are formulas, then so are $\neg\alpha$, $\alpha \wedge \beta$ and $\alpha \vee \beta$,
- if α is formula, then so are $\mathrm{EX}\alpha$, $\mathrm{EG}\alpha$ and $\mathrm{EU}(\alpha, \beta)$,
- if α is formula, then so are $\overline{\mathcal{O}}_i\alpha$, and $\overline{\mathrm{K}}_i\alpha$, for $i \in \mathcal{A}$.

Intuitively, E means there exists a computation, Xα is true in a computation if α is true at the second state of the computation, U(α, β) is true in a computation if β is true at some state on the computation and always earlier α holds, and Gα is true in a computation if α is true at all the states of the computation. We use the indexed modal operator $\overline{\mathcal{O}}_i$ to represent the *correctly functioning circumstances of agent i*. The formula $\overline{\mathcal{O}}_i\alpha$ stands for "there is a state where agent i is functioning correctly, and in which α holds". We

refer to [10,11] for a discussion of this notion [8]. Moreover we use the indexed modality $\overline{\mathrm{K}}_i$ to represent the diamond of an epistemic operator for agent i [8]: $\overline{\mathrm{K}}_i\alpha$ stands for "agent i considers possible that α".

The derived basic modalities are defined as follows: $\mathrm{EF}\alpha \stackrel{def}{=} \mathrm{EU}(true, \alpha)$, $\mathrm{AX}\alpha \stackrel{def}{=} \neg\mathrm{EX}\neg\alpha$, $\mathrm{AF}\alpha \stackrel{def}{=} \neg\mathrm{EG}\neg\alpha$, $\mathrm{AR}(\alpha, \beta) \stackrel{def}{=} \neg\mathrm{EU}(\neg\alpha, \neg\beta)$, $\mathrm{AG}\alpha \stackrel{def}{=} \neg\mathrm{EF}\neg\alpha$, $\mathcal{O}_i\alpha \stackrel{def}{=} \neg\overline{\mathcal{O}}_i\neg\alpha$ for $i \in \mathcal{A}$, $\mathrm{K}_i\alpha \stackrel{def}{=} \neg\overline{\mathrm{K}}_i\neg\alpha$ for $i \in \mathcal{A}$. Moreover, $\alpha \rightarrow \beta \stackrel{def}{=} \neg\alpha \vee \beta$.

The logic ECTLKD is the restriction of CTLKD such that the negation can be applied only to elements of $\mathcal{PV}$, i.e., $\neg\alpha$ is replaced by $\neg p$ in Definition 3.1.

The logic ACTLKD is the restriction of CTLKD such that its language is defined as $\{\neg\varphi \mid \varphi \in \mathrm{ECTLKD}\}$. It is easy to see that ACTLKD consists of the temporal formulas of the form: $\mathrm{AX}\alpha$, $\mathrm{AR}(\alpha, \beta)$, $\mathrm{AF}\alpha$, $\mathrm{K}_i\alpha$ and $\mathcal{O}_i\alpha$.

Definition 3.2 [Semantics of CTLKD]
Let M be a model, s be a state, and α, β be formulas of CTLKD. $M, s \models \alpha$ denotes that α is true at the state s in the model M. M is omitted, if it is implicitly understood. The relation $\models$ is defined inductively as follows:

$$
\begin{array}{lll}
s \models p & \text{iff} & p \in \mathcal{V}(s), \\
s \models \neg\alpha & \text{iff} & s \not\models \alpha, \\
s \models \alpha \vee \beta & \text{iff} & s \models \alpha \ \text{ or } \ s \models \beta, \\
s \models \alpha \wedge \beta & \text{iff} & s \models \alpha \ \text{ and } \ s \models \beta, \\
s \models \mathrm{EX}\alpha & \text{iff} & (\exists \pi \in \Pi(s))\ \pi(1) \models \alpha, \\
s \models \mathrm{EG}\alpha & \text{iff} & (\exists \pi \in \Pi(s))(\forall m \geq 0)\ \pi(m) \models \alpha, \\
s \models \mathrm{EU}(\alpha, \beta) & \text{iff} & (\exists \pi \in \Pi(s))(\exists m \geq 0)\ [\pi(m) \models \beta \text{ and } (\forall j < m)\ \pi(j) \models \alpha], \\
s \models \overline{\mathcal{O}}_i\alpha & \text{iff} & \exists s' \in \mathcal{DS}\ (sR_i^O s' \ and \ s' \models \alpha), \\
s \models \overline{\mathrm{K}}_i\alpha & \text{iff} & \exists s' \in \mathcal{DS}\ (sR_i^K s' \ and \ s' \models \alpha).
\end{array}
$$

Definition 3.3 [Validity] A CTLKD formula φ is valid in a model $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ (denoted $M \models \varphi$) iff $M, \iota \models \varphi$, i.e., φ is true at the initial state of the model M.

[8] Note that the operator $\overline{\mathcal{O}}_i$ is there referred to as $\mathcal{P}_i$.

4 Bounded Semantics for ECTLKD

In this section we give a *bounded semantics* for ECTLKD in order to define the *bounded model checking problem* for ECTLKD, and to translate it subsequently into a satisfiability problem. This formalism is an extension of the one presented in [14].

Definition 4.1 [$k-$model]
Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model and $k \in \mathbb{N}_+$. A tuple $M_k = (\mathcal{DS}, \iota, P_k, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ is a $k-$model for M, where P_k is the set of all the k-computations of M, i.e., $P_k = \bigcup_{s \in \mathcal{DS}} \Pi_k(s)$.

Satisfaction for the temporal operators in the bounded case depends on whether or not the computation π defines a loop, i.e., whether $loop(\pi) \neq \emptyset$, where *loop* is defined below.

Definition 4.2 [loop]
Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model, $k \in \mathbb{N}_+$ be a bound, $P_k = \bigcup_{s \in \mathcal{DS}} \Pi_k(s)$, and $\pi \in P_k$. The function $loop : P_k \to 2^{\{0,\ldots,k\}}$ is defined as follows:

$$loop(\pi) \;=\; \{l \;\mid\; 0 \leq l \leq k \; and \; (\pi(k),\; \pi(l)) \in T\}$$

Note that the interpretation of the temporal modalities on bounded semantics is different from the one of Definition 3.2.

Definition 4.3 [Bounded semantics]
Let M_k be a $k-$model and α, β be ECTLKD formulas. $M_k, s \models \alpha$ denotes that α is true at the state s of M_k. M_k is omitted if it is clear from the context. The relation $\models$ is defined inductively as follows:

$$
\begin{array}{lll}
s \models p & \text{iff} & p \in \mathcal{V}(s), \text{ for } p \in \mathcal{PV}, \\
s \models \neg p & \text{iff} & p \notin \mathcal{V}(s), \text{ for } p \in \mathcal{PV}, \\
s \models \alpha \vee \beta & \text{iff} & s \models \alpha \text{ or } s \models \beta, \\
s \models \alpha \wedge \beta & \text{iff} & s \models \alpha \text{ and } s \models \beta, \\
s \models \mathrm{EX}\alpha & \text{iff} & (\exists \pi \in \Pi_k(s))\ \pi(1) \models \alpha, \\
s \models \mathrm{EG}\alpha & \text{iff} & (\exists \pi \in \Pi_k(s))(\forall 0 \leq j \leq k)(\pi(j) \models \alpha \text{ and } \mathit{loop}(\pi) \neq \emptyset), \\
s \models \mathrm{EU}(\alpha, \beta) & \text{iff} & (\exists \pi \in \Pi_k(s))(\exists 0 \leq j \leq k)(\pi(j) \models \beta \text{ and} \\
 & & (\forall 0 \leq i < j)\pi(i) \models \alpha), \\
s \models \overline{\mathcal{O}}_i\alpha & \text{iff} & (\exists \pi \in \Pi_k(\iota))(\exists 0 \leq j \leq k)(\pi(j) \models \alpha \text{ and } sR_i^O\pi(j)), \\
s \models \overline{\mathrm{K}}_i\alpha & \text{iff} & (\exists \pi \in \Pi_k(\iota))(\exists 0 \leq j \leq k)(\pi(j) \models \alpha \text{ and } sR_i^K\pi(j)).
\end{array}
$$

The above extends to deontic modalities the bounded semantics of [13,14]. As in [13] we note that the given Definition 2.1, the relations for the operator $\overline{\mathcal{O}}_i$ used above are constructed on the basis of the *internal structure of the global states* of the system (i.e., they are defined on the basis of the local states of the agents), and not by means of an ad-hoc construction by the modeller of the system. Note also that while the conditions for the temporal components require the states to be reachable from the state in consideration, this is not the case for operators $\overline{\mathcal{O}}_i$ and $\overline{\mathrm{K}}_i$, where we consider whether or not there is *a* computation from the initial state that results in a state that is related for agent i from the global state under consideration. This guarantees reachability of such a state and corresponds to the usual interpretation of the modalities in the non-bounded model.

The theoretical results proved in [13] for CTLK can easily be extended for CTLKD.

Definition 4.4 [Validity for Bounded Semantics] An ECTLKD formula φ is valid in a k-model M_k (denoted $M \models_k \varphi$) iff $M_k, \iota \models \varphi$.

Next, we describe how the model checking problem ($M \models \varphi$) can be reduced to the bounded model checking problem ($M \models_k \varphi$).

Lemma 4.5 *Let M be a model, s be a state of M, and φ be an* ECTLKD *formula. Then, the following two conditions hold:*

a) $M_k, s \models \varphi$ implies $M_l, s \models \varphi$, for $l \geq k$,

b) $M_k, s \models \varphi$ implies $M, s \models \varphi$.

Proof. Straightforward by induction on the length of φ. □

Lemma 4.6 *Let M be a model, φ be an* ECTLKD *formula, s be a state of M, and $k = |M|$. If $M, s \models \varphi$, then $M_k, s \models \varphi$.*

Proof. By induction on the length of φ. The lemma follows directly for the propositional variables and their negations.

Next, assume that the hypothesis holds for all the proper sub-formulas of φ. If φ is equal to either $\alpha \wedge \beta$ or $\alpha \vee \beta$, then it is easy to check that the lemma holds. Consider φ to be of the following forms:

- $\varphi = \mathrm{EX}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{EU}(\alpha, \beta)$. By induction hypothesis — see [14] page 139.
- $\varphi = \overline{\mathcal{O}}_i\alpha$. By definition, there is a state s' in M such that $l_i(s') \in \mathcal{G}_i$ and $M, s' \models \alpha$. By the inductive assumption, we have that $M_k, s' \models \alpha$. Since s' is reachable, it is reachable from ι in at most k steps as $k = |M|$. Thus, there is a $k-$computation $\pi \in P_k$ such that $\pi(0) = \iota$ and $\pi(i) = s'$ for some $i \leq k$. So, we have $M_k, s \models \overline{\mathcal{O}}_i\alpha$.
- $\varphi = \overline{\mathrm{K}}_i\alpha$. By definition, there is a state s' in M such that $l_i(s) = l_i(s')$ and $M, s' \models \alpha$. By the inductive assumption, we have that $M_k, s' \models \alpha$. Since s' is reachable, it is reachable from ι in at most k steps as $k = |M|$. Thus, there is a $k-$computation $\pi \in P_k$ such that $\pi(0) = \iota$ and $\pi(i) = s'$ for some $i \leq k$. So, we have $M_k, s \models \overline{\mathrm{K}}_i\alpha$.

□

In this setting we can prove that in some circumstances satisfiability in the $|M|$-bounded semantics is equivalent to the unbounded one.

Theorem 4.7 *Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model, φ be an* ECTLKD *formula and $k = |M|$. Then, $M \models \varphi$ iff $M \models_k \varphi$.*

Proof. Straightforward from Lemma 4.5 and Lemma 4.6 above. □

Given that we reasoned on a bounded model of size $|M|$ there is nothing surprising about the results above. The rationale behind the method is that for particular examples checking satisfiability of a formula can be done on a small fragment of the model.

5 The BMC algorithm for ECTLKD

In this section we present a Bounded Model Checking (BMC) method for ECTLKD. This is an extension of the method appearing in [13,14]. This construction first appeared in [14], and was then extended in [13] for the CTLK case.

Definition 5.1 Let $M_k = (\mathcal{DS}, \iota, P_k, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a k-model of M. We say that a structure $M_k' = (\mathcal{DS}', \iota, P_k', R_1'^O, \ldots, R_n'^O, R_1'^K, \ldots, R_n'^K, \mathcal{V}')$ is a *submodel* of M_k if $P_k' \subseteq P_k$, $States(P_k') \subseteq \mathcal{DS}' \subseteq \mathcal{DS}$, $R_i'^O = R_i^O \cap (\mathcal{DS}' \times \mathcal{DS}')$, for $i \in \mathcal{A}$, $R_i'^K = R_i^K \cap (\mathcal{DS}' \times \mathcal{DS}')$, for $i \in \mathcal{A}$, and $\mathcal{V}' = \mathcal{V}|_{\mathcal{DS}'}$, where $States(P_k')$ defines the set of states reached in all computations in P_k', and $\mathcal{V}|_{\mathcal{DS}'}$ denotes the restriction of the interpretation function $\mathcal{V}$ to $\mathcal{DS}'$, a subset of $\mathcal{DS}$ (upon which $\mathcal{V}$ is defined).

For technical reasons we allow for having states in $\mathcal{DS}'$, which may not be reached in P_k', but obviously all the states of $\mathcal{DS}'$ are reachable in M_k as $\mathcal{DS}' \subseteq \mathcal{DS}$. The bounded semantics of ECTLKD over submodels M_k' can still be defined as for M_k (see Def. 4.3). Our present aim is give a bound for the number of k-computations in M_k' such that the validity of φ in M_k is equivalent to the validity of φ in M_k'.

Definition 5.2 Define a function $f_k : \mathcal{FORM} \to \mathbb{N}$ as follows:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\mathrm{EG}\alpha) = (k+1) \cdot f_k(\alpha) + 1$,
- $f_k(\mathrm{EU}(\alpha, \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(Y\alpha) = f_k(\alpha) + 1$, for $Y \in \{\mathrm{EX}, \overline{\mathrm{K}}_i, \overline{\mathcal{O}}_i\}$.

The function f_k determines the number of k-computations of a submodel M_k' sufficient for checking an ECTLKD formula. Here we take this bound as given, but we provide a proof of the adequacy of this in the next section.

The main idea of the BMC method is that we can check φ over M_k by checking the satisfiability of a propositional formula $[M, \varphi]_k = [M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$, where the first conjunct represents (part of) the model under consideration, and the second a number of constraints that must be satisfied on M_k for φ to be satisfied. Once this translation is defined, checking satisfiability of an ECTLKD formula can be done by means of a SAT-checker. Although from a theoretical point of view the complexity of this operation is no easier, in practice the efficiency of modern SAT-checkers makes the process worthwhile in many instances. In this process, an important decision to take is the size k of the truncation; there are heuristics that can be developed for particular classes of examples [2]. A trivial mechanism, for instance, would be to start with $k := 1$, test satisfiability for the translation, and increase k by one either until $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$ becomes satisfiable or k reaches $|M|$.

Definition 5.3 BMC algorithm for ECTLKD:

(1) Let $\varphi := \neg\psi$ (where ψ is an ACTLKD formula).
(2) Set $k := 1$.
(3) Select the $k-$model M_k.
(4) Select the submodels M'_k of M_k with $|P_k{}'| \leq f_k(\varphi)$.
(5) Translate the transition relation of all the submodels M'_k of M_k into a propositional formula $[M^{\varphi,\iota}]_k$.
(6) Translate φ over all M'_k into a propositional formula $[\varphi]_{M_k}$.
(7) Check the satisfiability of $[M,\varphi]_k := [M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$.
(8) If $[M,\varphi]_k$ is satisfiable, then return $M \models \varphi$ (i.e., $M \not\models \psi$), else set $k := k+1$.
(9) If $k = |M|+1$, then return $M \not\models \varphi$ (i.e., $M \models \psi$) else go to 3.

Now, we give details of this translation. We begin with an encoding of the transitions in the interpreted system under consideration. Recall that the set of reachable global states is $\mathcal{DS} \subseteq \prod_{i=1}^{n} L_i \times L_e$, where $L_i \supseteq \mathcal{G}_i$ for each agent $i \in \mathcal{A}$, and $L_e \supseteq \mathcal{G}_e$ for the environment. We assume that $L_i = \mathcal{G}_i \cup \mathcal{R}_i \subseteq \{0,1\}^{g_i} \times \{0,1\}^{r_i}$, where $g_i = \lceil log_2(|\mathcal{G}_i|)\rceil$, $r_i = \lceil log_2(|\mathcal{R}_i|)\rceil$, and $L_e = \mathcal{G}_e \cup \mathcal{R}_e \subseteq \{0,1\}^{g_e} \times \{0,1\}^{r_e}$, where $g_e = \lceil log_2(|\mathcal{G}_e|)\rceil$, $r_e = \lceil log_2(|\mathcal{R}_e|)\rceil$. Let $g_1 + r_1 + \ldots + g_n + r_n + g_e + r_e = m$. Then, each global state $s = (l_1,\ldots,l_n,l_e) = (s[1],\ldots,s[m])$ can be represented by $w = (w[1],\ldots,w[m])$ (which we shall call a *global state variable*), where each $w[i]$ for $i = 1,\ldots,m$ is a propositional variable. Notice that we distinguish between global states being sequences of binary digits and their representations in terms of propositional variables $w[i]$. A finite sequence $(w_0,\ldots,w_k)$ of global state variables is called a *symbolic $k-$path*. In general we shall need to consider not just one but a number of symbolic $k-$paths. This number depends on the formula φ under investigation, and it is returned as the value $f_k(\varphi)$ of the function f_k. We refer to [14] for more details. To construct $[M,\varphi]_k$, we first define a propositional formula $[M^{\varphi,\iota}]_k$ that defines the $f_k(\varphi)$ symbolic k-paths to be valid k-computations of M_k. For $j \in \{1,\ldots,f_k(\varphi)\}$, the j-th symbolic $k-$computation is denoted as $w_{0,j},\ldots,w_{k,j}$, where $w_{i,j}$ for $i \in \{0,\ldots,k\}$ are global state variables.

Let $\mathcal{SV}$ be a set of state variables, $\mathcal{SF}$ be a set of propositional formulas over $\mathcal{SV}$, and let $lit : \{0,1\} \times \mathcal{SV} \to \mathcal{SF}$ be a function defined as follows: $lit(0,p) = \neg p$ and $lit(1,p) = p$. Moreover, let $green_i : \mathcal{SV}^m \to \mathcal{SV}^{g_i}$, for $i = 1,\ldots,n,e$ be a function which returns the sequence of state variables encoding the green states of the i-th agent or environment, and let Idx_i and Idx_e be sets of the indexes of the bits of the local states of each agent i and environment in the global states. Furthermore, let w,v be global state

variables. We define the following propositional formulas:

- $I_s(w) := \bigwedge_{i=1}^{m} lit(s[i], w[i])$.
  This formula encodes the state s of the model, i.e., $s[i] = 1$ is encoded by $w[i]$, and $s[i] = 0$ is encoded by $\neg w[i]$.
- $p(w)$ is a formula over $w[1], \ldots, w[m]$, which is true for a valuation $(s_1, \ldots, s_m) \in \{0,1\}^m$ of $(w[1], \ldots, w[m])$ iff $p \in \mathcal{V}((s_1, \ldots, s_m))$, where $p \in \mathcal{PV}$.
  This formula encodes a proposition $p \in \mathcal{PV}$.
- $H(w, v) := \bigwedge_{i=1}^{m} w[i] \Leftrightarrow v[i]$.
  This formula represents logical equivalence between global state encodings, representing the fact that they represent the same state.
- $\mathrm{H}_{\mathrm{i}}^{\mathcal{O}}(w) := \bigvee_{l \in \mathcal{G}_i} (\bigwedge_{j=1}^{g_i} lit(l[j], green_i(w)[j]))$.
  This formula encodes an accessibility of a global state in which agent i is running correctly.
- $\mathrm{H}_{\mathrm{l}}^{\mathrm{K}}(w, v) := \bigwedge_{i \in Idx_l} w[i] \Leftrightarrow v[i]$.
  This formula represents logical equivalence between l-local state encodings, representing the fact that they represent the same local state, i.e., the local state in the two states is the same.
- $TR(w, v)$ is a formula over the propositions $w[1], \ldots, w[m]$, $v[1], \ldots, v[m]$ that is true for a valuation $(s_1, \ldots, s_m)$ of $(w[1], \ldots, w[m])$ and a valuation $(s'_1, \ldots, s'_m)$ of $(v[1], \ldots, v[m])$ iff $((s_1, \ldots, s_m), (s'_1, \ldots, s'_m)) \in T$.
- $L_{k,j}(l) := TR(w_{k,j}, w_{l,j})$.
  This formula encodes a backward loop connecting the k-th state to the l-th state in the symbolic $k-$computation j, for $0 \leq l \leq k$.

The propositional formula $[M^{\varphi,\iota}]_k$, representing the transitions in the k-model, is given by the following definition.

Definition 5.4 [Unfolding of Transition Relation]
Let $M = (\mathcal{DS}, \iota, T, R_1^O, \ldots, R_n^O, R_1^K, \ldots, R_n^K, \mathcal{V})$ be a model, $k \in \mathbb{N}_+$ be a bound, and φ be an ECTLKD formula. The propositional formula $[M^{\varphi,\iota}]_k$ is defined as follows:

$$[M^{\varphi,\iota}]_k := I_\iota(w_{0,0}) \wedge \bigwedge_{j=1}^{f_k(\varphi)} \bigwedge_{i=0}^{k-1} TR(w_{i,j}, w_{i+1,j})$$

where $w_{0,0}$, and $w_{i,j}$ for $0 \leq i \leq k$, and $1 \leq j \leq f_k(\varphi)$ are global state variables. $[M^{\varphi,\iota}]_k$ encodes the initial state ι by $w_{0,0}$ and constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-computations in M_k.

The next step of the algorithm consists in translating an ECTLKD formula φ into a propositional formula.

Definition 5.5 [Translation of ECTLKD formulas] Let a model M_k with initial state ι, and an ECTLKD formula φ be given. We inductively define the translation of φ at state $w_{m,n}$ into the propositional formula $[\varphi]_k^{[m,n]}$ as follows:

$$
\begin{array}{lcl}
[p]_k^{[m,n]} & := & p(w_{m,n}),\\
[\neg p]_k^{[m,n]} & := & \neg p(w_{m,n}),\\
[\alpha \wedge \beta]_k^{[m,n]} & := & [\alpha]_k^{[m,n]} \wedge\ [\beta]_k^{[m,n]},\\
[\alpha \vee \beta]_k^{[m,n]} & := & [\alpha]_k^{[m,n]} \vee\ [\beta]_k^{[m,n]},\\
[\mathrm{EX}\alpha]_k^{[m,n]} & := & \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \ \wedge\ [\alpha]_k^{[1,i]}\Big),\\
[\mathrm{EG}\alpha]_k^{[m,n]} & := & \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \ \wedge\ (\bigvee_{l=0}^{k} L_{k,i}(l)) \ \wedge\ \bigwedge_{j=0}^{k} [\alpha]_k^{[j,i]}\Big),\\
[\mathrm{EU}(\alpha,\beta)]_k^{[m,n]} & := & \bigvee_{i=1}^{f_k(\varphi)} \Big(H(w_{m,n}, w_{0,i}) \ \wedge\ \bigvee_{j=0}^{k} ([\beta]_k^{[j,i]} \ \wedge\ \bigwedge_{t=0}^{j-1} [\alpha]_k^{[t,i]})\Big),\\
[\overline{\mathcal{O}}_l\alpha]_k^{[m,n]} & := & \bigvee_{i=1}^{f_k(\varphi)} \Big(I_\iota(w_{0,i}) \ \wedge\ \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \ \wedge\ \mathrm{H}_{\mathrm{l}}^{\mathcal{O}}(w_{j,i}))\Big),\\
[\overline{\mathrm{K}}_l\alpha]_k^{[m,n]} & := & \bigvee_{i=1}^{f_k(\varphi)} \Big(I_\iota(w_{0,i}) \ \wedge\ \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \ \wedge\ \mathrm{H}_{\mathrm{l}}^{\mathrm{K}}(w_{m,n}, w_{j,i}))\Big).
\end{array}
$$

The meaning of the translations above can be intuitively reconstructed from the definition of propositional formulas presented earlier. For example, the formula $[\mathrm{EX}\alpha]_k^{[m,n]}$ expresses the condition that there exists a sub-path starting from $w_{m,n}$ in which the first point $w_{0,i}$ in this computation satisfies α. For $[\overline{\mathcal{O}}_l\alpha]_k^{[m,n]}$ we insist on the existence of a point $w_{j,i}$ in which agent l is in a green local state, and that it is accessible from the initial state by some computation. For $[\overline{\mathrm{K}}_l\alpha]_k^{[m,n]}$ we insist on the existence of a point $w_{j,i}$ in which agent l is in the same local state, and that it is accessible from the initial state by some computation.

Given the translations above, we can now check φ over M_k by checking the satisfiability of the propositional formula $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$, where $[\varphi]_{M_k} = [\varphi]_k^{[0,0]}$. The translation presented above is shown to be correct and complete in the next section.

6 Correctness of the translation

In this section we prove the correctness of the translation of the model checking problem into the SAT-problem as given by Definition 5.4.

Lemma 6.1 *$M_k, s \models \varphi$ iff there is a submodel M_k' of M_k with $|P_k'| \leq f_k(\varphi)$ such that $M_k', s \models \varphi$.*

Proof. (=>) By structural induction on φ. The lemma follows directly for the propositional variables and their negations.

Assume that the hypothesis holds for all the proper sub-formulas of φ.

- $\varphi = \alpha \vee \beta \mid \alpha \wedge \beta$. Straightforward.
- $\varphi = \mathrm{EX}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{EU}(\alpha, \beta)$. By induction hypothesis — see [14] page 143.
- Let $\varphi = \overline{\mathcal{O}}_i\alpha$. If $M_k, s \models \overline{\mathcal{O}}_i\alpha$, then by definition: $(\exists \pi \in P_k)(\pi(0) = \iota$ and $\exists_{0 \leq j \leq k}(sR_i^O\pi(j))$ and $\pi(j) \models \alpha))$. By the inductive assumption there is a submodel $M_k' = (\mathcal{DS}', \iota, P_k', R_1'^O, \ldots, R_n'^O, R_1'^K, \ldots, R_n'^K, \mathcal{V}')$ of M_k such that $|P_k'| \leq f_k(\alpha)$ and $M_k', \pi(j) \models \alpha$.
  Consider a submodel $M_k'' = (\mathcal{DS}'', P_k'', R_1''^O, \ldots, R_n''^O, R_1''^K, \ldots, R_n''^K, \iota, \mathcal{V}'')$ of M_k, where $P_k'' = P_k' \cup \{\pi\}$ and $\mathcal{DS}'' = States(P_k'') \cup \{s\}$. Since π belongs to P_k'', by the construction of M_k'' and the definition of the bounded semantics, we have that $M_k'', s \models \overline{\mathcal{O}}_i\alpha$ and $|P_k''| \leq f_k(\varphi) = f_k(\alpha) + 1$.
- Let $\varphi = \overline{\mathrm{K}}_i\alpha$. If $M_k, s \models \overline{\mathrm{K}}_i\alpha$, then by definition:
  $(\exists \pi \in P_k)(\pi(0) = \iota$ and $\exists_{0 \leq j \leq k}(sR_i^K\pi(j))$ and $\pi(j) \models \alpha))$. By induction there is a submodel $M_k' = (\mathcal{DS}', \iota, P_k', R_1'^O, \ldots, R_n'^O, R_1'^K, \ldots, R_n'^K, \mathcal{V}')$ of M_k such that $|P_k'| \leq f_k(\alpha)$ and $M_k', \pi(j) \models \alpha$.
  Consider a submodel $M_k'' = (\mathcal{DS}'', \iota, P_k'', R_1''^O, \ldots, R_n''^O, R_1''^K, \ldots, R_n''^K, \mathcal{V}'')$ of M_k, where $P_k'' = P_k' \cup \{\pi\}$ and $\mathcal{DS}'' = States(P_k'') \cup \{s\}$. Since π belongs to P_k'', by the construction of M_k'' and the definition of the bounded semantics, we have that $M_k'', s \models \overline{\mathrm{K}}_i\alpha$ and $|P_k''| \leq f_k(\varphi) = f_k(\alpha) + 1$.

(<=) The proof is straightforward. □

From Lemma 6.1 we can now derive the following.

Corollary 6.2 *$M \models_k \varphi$ iff there is a submodel M_k' of M_k with $|P_k'| \leq f_k(\varphi)$ such that $M_k', \iota \models \varphi$.*

Proof. It follows from Definition 4.4, and Lemma 6.1, by using $s = \iota$. □

Lemma 6.3 *For each state s of M, the following condition holds: $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ is satisfiable iff there is a submodel M_k' of M_k with $|P_k'| \leq f_k(\varphi)$ such that $M_k', s \models \varphi$.*

Proof. (=>) Let $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ be satisfiable. By the definition of the translation, the propositional formula $[\varphi]_{M_k}$ encodes all the sets of $k-$computations of size $f_k(\varphi)$ which satisfy the formula φ. By the definition of the unfolding of the transition relation, the propositional formula $[M^{\varphi,s}]_k$ encodes $f_k(\varphi)$ symbolic k-paths to be valid $k-$computations of M_k. Hence, there is a set of $k-$computations in M_k, which satisfies the formula φ, of size smaller or equal to $f_k(\varphi)$. Thus, we conclude that there is a submodel M_k' of M_k with

$|P_k'| \leq f_k(\varphi)$ and $M_k', s \models \varphi$. The actual definition of M_k' can be reconstructed from Definition 5.5 and Definition 5.4.

(<=) The proof is by induction on the length of φ. The lemma follows directly for the propositional variables and their negations. Consider the following cases:

- For $\varphi = \alpha \vee \beta, \alpha \wedge \beta$ or the temporal operators the proof is like in [14].
- Let $\varphi = \overline{\mathcal{O}}_l\alpha$. If $M_k', s \models \overline{\mathcal{O}}_l\alpha$ with $|P_k'| \leq f_k(\overline{\mathcal{O}}_l\alpha)$, then by Definition 4.3 we have that there is a $k-$computation π such that $\pi(0) = \iota$ and $(\exists j \leq k)\ sR_l^{\mathcal{O}}\pi(j)$ and $M_k', \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]_k^{[0,0]} \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formula $I_\iota(w_{0,ii})$. Therefore, by the construction above, it follows that the propositional formula $I_\iota(w_{0,ii}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,ii]} \wedge \mathrm{H}_\mathrm{l}^{\mathcal{O}}(w_{j,ii})) \wedge [M^{\overline{\mathcal{O}}_l\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable: $\bigvee_{1\leq i\leq f_k(\overline{\mathcal{O}}_l\alpha)} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \mathrm{H}_\mathrm{l}^{\mathcal{O}}(w_{j,i})) \wedge [M^{\overline{\mathcal{O}}_l\alpha,s}]_k\Big)$.
  Hence, by the definition of the translation of an ECTLKD formula, the above formula is equal to the propositional formula $[\overline{\mathcal{O}}_l\alpha]_k^{[0,0]} \wedge [M^{\overline{\mathcal{O}}_l\alpha,s}]_k$.
- Let $\varphi = \overline{\mathrm{K}}_l\alpha$. If $M_k', s \models \overline{\mathrm{K}}_l\alpha$ with $|P_k'| \leq f_k(\overline{\mathrm{K}}_l\alpha)$, then by Definition 4.3 we have that there is a $k-$computation π such that $\pi(0) = \iota$ and $(\exists j \leq k)\ sR_l^{K}\pi(j)$ and $M_k', \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]_k^{[0,0]} \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formula $I_\iota(w_{0,ii})$. Therefore, by the construction above, it follows that the propositional formula $I_\iota(w_{0,ii}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,ii]} \wedge \mathrm{H}_\mathrm{l}^{\mathrm{K}}(w_{0,0}, w_{j,ii})) \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:
  $\bigvee_{1\leq i\leq f_k(\overline{\mathrm{K}}_l\alpha)} \Big(I_\iota(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \mathrm{H}_\mathrm{l}^{\mathrm{K}}(w_{0,0}, w_{j,i})) \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k\Big)$.
  Hence, by the definition of the translation of an ECTLKD formula, the above formula is equal to the propositional formula $[\overline{\mathrm{K}}_l\alpha]_k^{[0,0]} \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k$.

□

Theorem 6.4 *Let M be a model, M_k be a $k-$model of M, and φ be an* ECTLKD *formula. Then, $M \models_k \varphi$ iff $[\varphi]_{M_k} \wedge [M^{\varphi,\iota}]_k$ is satisfiable.*

Proof. Follows from Lemmas 6.1 and 6.3. □

Corollary 6.5 *$M \models_k \neg\varphi$ iff $[\varphi]_{M_k} \wedge [M^{\varphi,\iota}]_k$ is unsatisfiable for $k = |M|$.*

This concludes our analysis of the translation technique. We now give an example to demonstrate how it can be put into practice.

7 Model checking the bit transmission problem with faults

The bit-transmission problem [8] involves two agents, a *sender* $\mathfrak{S}$, and a *receiver* $\mathfrak{R}$, communicating over a possibly faulty communication channel. $\mathfrak{S}$ wants to communicate some information—the value of a bit for the sake of the example—to $\mathfrak{R}$. One protocol to achieve this is as follows [8]. $\mathfrak{S}$ immediately starts sending the bit to $\mathfrak{R}$, and continues to do so until it receives an acknowledgement from $\mathfrak{R}$. $\mathfrak{R}$ does nothing until it receives the bit; from then on it sends acknowledgements of receipt to $\mathfrak{S}$. $\mathfrak{S}$ stops sending the bit to $\mathfrak{R}$ when it receives an acknowledgement. Note that $\mathfrak{R}$ will continue sending acknowledgements even after $\mathfrak{S}$ has received its acknowledgement. Intuitively, $\mathfrak{S}$ will know for sure that $\mathfrak{R}$ received the bit when it gets an acknowledgement from $\mathfrak{R}$. $\mathfrak{R}$, on the other hand, will never be able to know whether its acknowledgement has been received since $\mathfrak{S}$ does not answer the acknowledgement. We refer to [9] for further discussion.

In this section we are interested in applying the machinery of bounded model checking to verify a version of the scenario above where one agent does not operate as it is supposed to. This version of the scenario was first described in [10]. In particular we examine in detail only the possibility that $\mathfrak{R}$ is faulty [9]. Specifically, we shall consider in this section the possibility that $\mathfrak{R}$ may send acknowledgements without having received the bit. This is a simple example of an agent not following its specification. This scenario can be analysed by means of deontic interpreted systems. We report here briefly part of the analysis that was conducted in [10], and then proceed to model check the example.

There are three active components in the scenario: a sender, a receiver, and a communication channel. In line with the spirit of the formalism of (deontic) interpreted systems, it is convenient to see sender and receiver as agents, and the communication channel as the environment. Each of these can be modelled by considering their local states [10]. For the sender $\mathfrak{S}$, it is enough to consider four possible local states and since we are not admitting the possibility of faults, its local states are all green. They represent the value of the bit that $\mathfrak{S}$ is attempting to transmit, and whether or not $\mathfrak{S}$ has received an acknowledgement from $\mathfrak{R}$. We thus have: $L_{\mathfrak{S}} = \mathcal{G}_{\mathfrak{S}} = \{0,\ 1,\ 0\text{-}ack,\ 1\text{-}ack\}, \quad \mathcal{R}_{\mathfrak{S}} = \emptyset$. For the environment it is enough to consider a singleton: $L_{\mathfrak{E}} = \{\cdot\}$. More-

[9] The possibility that $\mathfrak{S}$ is faulty, and other combinations of faulty $\mathfrak{R}$, $\mathfrak{S}$ and $\mathfrak{E}$, can be treated in similar fashion.

[10] Recall that, in order to apply the machinery of deontic interpreted systems we have to split the set of local states of agent i into two disjoint sets: green ($\mathcal{G}_i$) and red ($\mathcal{R}_i$), representing correct and incorrect functioning behaviour respectively.

over, we assume that all local states of the environment are green, so we have: $L_{\mathfrak{E}} = \mathcal{G}_{\mathfrak{E}}, \quad \mathcal{R}_{\mathfrak{E}} = \emptyset$. It remains to model the local states of the receiver $\mathfrak{R}$. Six different local states are enough to capture the state of $\mathfrak{R}$: the value of the received bit, the circumstance in which no bit has been received yet (represented by ϵ), the circumstance in which $\mathfrak{R}$ has sent an acknowledgement without having received the value of the bit (denoted by ϵ-*ack*), and the circumstance in which $\mathfrak{R}$ has sent an acknowledgement having received the value of the bit (represented by 0-*ack* and 1-*ack*). It remains to determine which local states of $\mathfrak{R}$ should be classified as red, and which as a green. This depends on how we interpret the *ack* component. We can view the *ack* as a proof that one faulty acknowledgement was sent before the value of the bit was received. On this reading we obtain the following partition of the set $L_{\mathfrak{R}} = \{0, 1, \epsilon, 0\text{-}ack, 1\text{-}ack, \epsilon\text{-}ack\}$ of $\mathfrak{R}$'s local states: $\mathcal{G}_{\mathfrak{R}} = \{0, 1, \epsilon\}$, and $\mathcal{R}_{\mathfrak{R}} = \{\epsilon\text{-}ack$,0-ack,1-ack$\}$.

The set of actions available to the agents are as follows: $Act_{\mathfrak{S}} = \{sendbit, \lambda\}$, $Act_{\mathfrak{R}} = \{sendack, \lambda\}$, where λ stands for no action ('no-op'). The actions $Act_{\mathfrak{E}}$ for the environment correspond to the transmission of messages between $\mathfrak{S}$ and $\mathfrak{R}$ on the unreliable communication channel. We will assume that the communication channel can transmit messages in both directions simultaneously, and that a message travelling in one direction can get through while a message travelling in the opposite direction is lost. The set of actions for the environment is $Act_{\mathfrak{E}} = \{\leftrightarrow, \rightarrow, \leftarrow, -\}$, where $\leftrightarrow$ represents the action in which the channel transmits any message successfully in both directions, $\rightarrow$ that it transmits successfully from $\mathfrak{S}$ to $\mathfrak{R}$ but loses any message from $\mathfrak{R}$ to $\mathfrak{S}$, $\leftarrow$ that it transmits successfully from $\mathfrak{R}$ to $\mathfrak{S}$ but loses any message from $\mathfrak{S}$ to $\mathfrak{R}$, and $-$ that it loses any messages sent in either direction.

The protocols the agents are running are as follows:

- $P_{\mathfrak{S}}(0) = P_{\mathfrak{S}}(1) = \{sendbit\}$, $P_{\mathfrak{S}}(0\text{-}ack) = P_{\mathfrak{S}}(1\text{-}ack) = \{\lambda\}$,
- $P_{\mathfrak{R}}(0) = P_{\mathfrak{R}}(1) = \{sendack\}$, $P_{\mathfrak{R}}(\epsilon) = \{\lambda\}$,
  $P_{\mathfrak{R}}(0\text{-}ack) = P_{\mathfrak{R}}(1\text{-}ack) = \{sendack\}$, $P_{\mathfrak{R}}(\epsilon\text{-}ack) = \{\lambda, sendack\}$.
- $P_{\mathfrak{E}}(\cdot) = Act_{\mathfrak{E}} = \{\leftrightarrow, \rightarrow, \leftarrow, -\}$.

Note that if $\mathfrak{R}$ performs a faulty action there is no possibility of recovery. It should be straightforward to infer the transition system that is induced by the informal description of the scenario we considered above together with the local states and protocols defined above. We refer to [10] for further discussion.

We now encode the local states in binary form in order to use them in the model checking technique. Since the sender $\mathfrak{S}$ can be in 4 different local green states we shall need 2 bits to encode its state; we take: $(0, 0) = 0$, $(0, 1) = 1$, $(1, 0) = 0\text{-}ack$, $(1, 1) = 1\text{-}ack$. Since the receiver $\mathfrak{R}$ can be in 3 different

local green states and in 3 different local red states, we shall need 2 + 2 bits to encode its state; we take: $(1,0;0,0) = 0$, $(0,1;0,0) = 1$, $(0,0;0,0) = \epsilon$, $(0,0;1,0) = 0\text{-}ack$, $(1,1;0,0) = 1\text{-}ack$, $(1,1;1,0) = \epsilon\text{-}ack$. The modelling of the environment $\mathfrak{E}$ requires only one bit: $(0) = \cdot$

In view of this, a global state is modelled by a byte: $s = (s[1], s[2], s[3], s[4], s[5], s[6], s[7])$. For instance the initial state $\iota = (0, \epsilon, \cdot)$ is represented as a tuple of seven 0's. If we want to represent it in terms of propositional variables, we shall have to insist on the propositions encoding the state to be in the state of false. In other words, we would encode the initial state as follows: $I_\iota(w_{0,0}) = \bigwedge_{i=1}^{7} \neg w_{0,0}[i]$.

Let $\mathcal{PV} = \{\mathbf{bit = 0}, \mathbf{bit = 1}, \mathbf{recbit}, \mathbf{recack}\}$. We use the following interpretation for the proposition variables in $\mathcal{PV}$:

$(M,s) \models \mathbf{bit = 0}$ if $l_{\mathfrak{S}}(s) = 0$ or $l_{\mathfrak{S}}(s) = 0\text{-}ack$,

$(M,s) \models \mathbf{bit = 1}$ if $l_{\mathfrak{S}}(s) = 1$ or $l_{\mathfrak{S}}(s) = 1\text{-}ack$,

$(M,s) \models \mathbf{recbit}$ if $l_{\mathfrak{R}}(s) = 1$ or $l_{\mathfrak{R}}(s) = 0$ or $l_{\mathfrak{R}}(s) = 0\text{-}ack$ or $l_{\mathfrak{R}}(s) = 1\text{-}ack$,

$(M,s) \models \mathbf{recack}$ if $l_{\mathfrak{S}}(s) = 1\text{-}ack$ or $l_{\mathfrak{S}}(s) = 0\text{-}ack$.

Some properties we may be interested in checking for the example above are the following:

(i) $\mathrm{AG}\Big(\neg\mathbf{recack} \vee \mathrm{K}_{\mathfrak{S}}(\mathcal{O}_{\mathfrak{R}}(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 1})))\Big)$

(ii) $\mathcal{O}_{\mathfrak{R}}(\mathbf{recack} \wedge \neg(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 1})))$

(iii) $\mathrm{AG}\Big(\mathcal{O}_{\mathfrak{R}}(\mathrm{K}_{\mathfrak{S}}(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 1})))\Big)$

(iv) $\mathrm{AU}\Big(\mathcal{O}_{\mathfrak{R}}(\mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 0}) \vee \mathrm{K}_{\mathfrak{R}}(\mathbf{bit = 1})),\ \mathbf{recack}\Big)$

Property (i) says that forever in the future if an *ack* is received by $\mathfrak{S}$, then $\mathfrak{S}$ knows that in all the states where $\mathfrak{R}$ is functioning correctly, $\mathfrak{R}$ knows the value of the bit. Property (ii) states that in all the states where $\mathfrak{R}$ is functioning correctly $\mathfrak{S}$ has received an acknowledgement and $\mathfrak{R}$ does not know the value of the bit. Property (iii) says that forever in the future in all the states where $\mathfrak{R}$ is functioning correctly, $\mathfrak{S}$ knows that $\mathfrak{R}$ knows the value of the bit. Property (iv) says that at one point at the future an *ack* is received by $\mathfrak{S}$ and at all the preceding points in time in all states where $\mathfrak{R}$ was operating as intended $\mathfrak{R}$ knew the value of the bit.

The property (i) is true on the interpreted system in consideration, but the properties (ii), (iii) and (iv) are not. The formula (i) is an ACTLKD formula, so in order to check it we shall have to encode the whole model. We can do this in the BMC technique reported above, but, as mentioned already,

the benefits of BMC are most apparent when only a *fraction* of the model is generated. For example this happens in formulas (ii), (iii) and (iv) where we need to check validity of an ECTLKD formula in the model. For the purposes of this paper we check validity of the formula (iii). The negated formula is:

$$\varphi := \mathrm{EF}\Big(\overline{\mathcal{O}}_{\mathfrak{R}}(\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1})))\Big)$$

The translation for the propositions used in φ is as follows: $(\mathbf{bit}=\mathbf{0})(w)$ $:= (\neg w[1] \wedge \neg w[2]) \vee (w[1] \wedge \neg w[2])$, which means that $(\mathbf{bit}=\mathbf{0})$ holds at all the global states with the first local state equal to $(0,0)$ or $(1,0)$. $(\mathbf{bit}=\mathbf{1})(w)$ $:= (\neg w[1] \wedge w[2]) \vee (w[1] \wedge w[2])$, which means that $(\mathbf{bit}=\mathbf{1})$ holds at all the global states with the first local state equal to $(0,1)$ or $(1,1)$.

The translation for the equality of the $\mathfrak{R}$-local states is as follows: $\mathrm{H}^{\mathrm{K}}_{\mathfrak{R}}(w,v) = \bigwedge_{i=3}^{6} w[i] \Leftrightarrow v[i]$, and the translation of an accessibility of a global state in which $\mathfrak{R}$ is running correctly is as follows: $\mathrm{H}^{\mathcal{O}}_{\mathfrak{R}}(v) = (v[3] \wedge \neg v[4]) \vee (\neg v[3] \wedge v[4]) \vee (\neg v[3] \wedge \neg v[4]) \vee (\neg v[3] \wedge \neg v[4]) \vee (v[3] \wedge v[4])$. The translation of the equality of the $\mathfrak{S}$-local states is as follows: $\mathrm{H}^{\mathrm{K}}_{\mathfrak{S}}(w,v) = \bigwedge_{i=1}^{2} w[i] \Leftrightarrow v[i]$.

We calculate that $f_k(\varphi) = 5$ for all $k \in \mathbb{N}_+$ (see Definition 5.2), so we need to exploit five symbolic $k-$paths. To proceed with the translation, the first thing we need to translate is the initial state $\iota = (0, \epsilon, \cdot)$, where ι is binary represented by $(0,\ldots,0)$. With the representation above this will be encoded by the propositional formula $I_\iota(w_{0,j}) := \bigwedge_{i=1}^{7} \neg w_{0,j}[i]$, for $0 \leq j \leq 5$.

The next step is to translate the transitions $T(w_{i,j}, w_{i+1,j})$; for simplicity we report only on one transition for the case $k = 1$, and in particular only on the formula $T(w_{0,1}, w_{1,1})$ representing the first transition of the first path. The remaining formulas are $T(w_{0,2}, w_{1,2})$, $T(w_{0,3}, w_{1,3})$, $T(w_{0,4}, w_{1,4})$ and $T(w_{0,5}, w_{1,5})$.

To encode the whole example we should model all the transitions for all the k's starting from $k := 1$. We do not do it here. Let us now encode the formula φ we would like to check.

$[\varphi]_1^{[0,0]} :=$

$\bigvee_{i=1}^{5} \Big(H(w_{m,n}, w_{0,i}) \ \wedge \ \bigvee_{j=0}^{k} [\overline{\mathcal{O}}_{\mathfrak{R}}(\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1})))]_k^{[j,i]}\Big)$

Next:

$[\overline{\mathcal{O}}_{\mathfrak{R}}(\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1})))]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge$

$\bigvee_{j=0}^{k} (\mathrm{H}^{\mathcal{O}}_{\mathfrak{R}}(w_{j,i}) \ \wedge \ [\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1}))]_k^{[j,i]})\Big)$

Next:

$[\overline{\mathrm{K}}_{\mathfrak{S}}(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1}))]_k^{[j,i]} := \bigvee_{i=1}^{5} \Big(I_\iota(w_{0,i}) \wedge$

$\bigvee_{j=0}^{k} ([(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0}) \wedge \overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1}))]_k^{[j,i]} \ \wedge \ \mathrm{H}^{\mathrm{K}}_{\mathfrak{S}}(w_{m,n}, w_{j,i}))\Big)$

Next:
$[(\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0})\wedge\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1}))]_k^{[j,i]} = [\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0})]_k^{[j,i]}\wedge[\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1})]_k^{[j,i]}$
Next:
$[\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{0})]_k^{[j,i]} := \bigvee_{i=1}^{5}\left(I_\iota(w_{0,i})\wedge\bigvee_{j=0}^{k}([\neg(\mathbf{bit}=\mathbf{0})]_k^{[j,i]}\wedge \mathrm{H}^{\mathrm{K}}_{\mathfrak{R}}(w_{m,n},w_{j,i}))\right)$
$[\overline{\mathrm{K}}_{\mathfrak{R}}\neg(\mathbf{bit}=\mathbf{1})]_k^{[j,i]} := \bigvee_{i=1}^{5}\left(I_\iota(w_{0,i})\wedge\bigvee_{j=0}^{k}([\neg(\mathbf{bit}=\mathbf{1})]_k^{[j,i]}\wedge \mathrm{H}^{\mathrm{K}}_{\mathfrak{R}}(w_{m,n},w_{j,i}))\right)$
Next:
$[\neg(\mathbf{bit}=\mathbf{0})]_k^{[j,i]} := \neg(\mathbf{bit}=\mathbf{0})(w_{j,i})$ and $[\neg(\mathbf{bit}=\mathbf{1})]_k^{[j,i]} := \neg(\mathbf{bit}=\mathbf{1})(w_{j,i})$.

Checking that the bit transmission protocol satisfies the temporal deontic formula above can now be done by feeding a SAT solver with the propositional formula generated in this method. This would produce a solution, thereby proving that the propositional formula is satisfiable.

8 Conclusions

In this paper we extended the methodology of bounded model checking for CTLK, presented in [13] by adding the deontic notion of correct functioning behaviours of the agents. This notion was explored in [11,10].

Model checking of deontic interpreted system may also be performed by means of Ordered Binary Diagrams (OBDD). This was explored in [15,16].

Future work include an implementation of the algorithm presented here, a careful evaluation of experimental results to be obtained, and a comparison of the OBDD and SAT based model checking method for deontic interpreted systems.

References

[1] N. Amla, R. Kurshan, K. McMillan, and R. Medel. Experimental analysis of different techniques for bounded model checking. In *Proc. of TACAS'03*, volume 2619 of *LNCS*, pages 34–48. Springer-Verlag, 2003.

[2] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded Model Checking. In *Advances in Computers*, volume 58. Academic Press, 2003. pre-print.

[3] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[4] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[5] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Vardi. Benefits of bounded model checking at an industrial setting. In *Proc. of CAV'01*, volume 2102 of *LNCS*, pages 436–453. Springer-Verlag, 2001.

[6] D. Dennett. *The Intentional Stance*. MIT Press, 1987.

[7] E. A. Emerson and E. M. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[9] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.

[10] A. Lomuscio and M. Sergot. Violation, error recovery, and enforcement in the bit transmission problem. In *Proceedings of DEON'02*, London, May 2002. *Journal of Applied Logic*, 2:93-116, Elsevier, 2004.

[11] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75, 2003.

[12] K. McMillan. The SMV system. Technical Report CMU-CS-92-131, Carnegie-Mellon University, February 1992.

[13] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[14] W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.

[15] F. Raimondi and A. Lomuscio. Automatic verification of deontic interpreted systems by model checking via OBDD's. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI04)*, August 2004.

[16] F. Raimondi and A. Lomuscio. Symbolic model checking of multi-agent systems via OBDD's: an algorithm and its implementation. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, LNCS. Spinger Verlag, 2004. to appear.

[17] M. Wooldridge. *An introduction to multi-agent systems*. John Wiley, England, 2002.