Fundamenta Informaticae 62 1–20
IOS Press

From bounded to unbounded model checking for temporal epistemic logic[*]

M. Kacprzak
Białystok University of Technology
Institute of Mathematics and Physics
15-351 Białystok, ul. Wiejska 45A, Poland
mdkacprzak@wp.pl

A. Lomuscio
Department of Computer Science
King's College London, London WC2R 2LS, United Kingdom
alessio@dcs.kcl.ac.uk

W. Penczek[†]
Institute of Computer Science, PAS
01-237 Warsaw, ul. Ordona 21, Poland
penczek@ipipan.waw.pl

Abstract. This paper addresses the problem of verification of temporal epistemic properties of multi-agent systems by means of symbolic model checking. An overview of the technique of bounded model checking for temporal epistemic logic, and an analysis of some limitations of the method are provided. An extension of this technique called unbounded model checking to solve these limitations is explored. Similarities and differences of the two methods are explicitly exemplified by the analysis of a scenario in the two formalisms.

[*]The authors acknowledge support from the Polish National Committee for Scientifi c Research (a special grant supporting ALFEBIITE), the Nuffi eld Foundation (grant NAL/00690/G), and EPSRC (GR/S49353/01).
[†]Address for correspondence: Institute of Computer Science, PAS, 01-237 Warsaw, ul. Ordona 21, Poland
Also affi liated with: Podlasie Academy, Institute of Informatics, Siedlce, Poland

1. Introduction

Verification of multi-agent systems [19] has recently become an active subject of research. In particular, recent contributions [16, 17, 18] have focused on extending model checking tools and techniques usually employed for verification of reactive systems. Essentially, these study the satisfiability of a formula representing a property of interest in the model representing all the computations of the multi-agent system (MAS) under consideration. The iterative approaches to model checking (i.e., those based on the explicit enumeration of all the possible states of computation) are known to suffer from the state explosion problem. For this reason, methods based on symbolic representation are currently seen as the most promising. In particular, the methods based on BDD's [11], and SAT-checkers [1, 3] are constant focus of research. Verification via BDD's involves translating the model checking problem into operations on Boolean functions represented in a concise and canonical way, whereas SAT-checkers are used for testing satisfiability of formulas, encoding the model checking problem.

In the multi-agent paradigm particular emphasis is given to high-level concepts such as knowledge, desires, and intentions of the agents. Because of this, work in MAS verification has focused on the extension of traditional model checking methods to incorporate modalities describing information and motivational attitudes of agents [2]. In particular, model checking algorithms for standard temporal logics of discrete time (LTL [10] or branching time CTL [7]) have been extended by means of an epistemic modality [16, 17, 18]. Specifically, the authors of this paper have suggested the use of the method of *bounded model checking* (BMC) for verifying CTLK formulas describing temporal epistemic properties of a MAS [13, 9]. CTLK combines branching temporal operators with epistemic ones like individual knowledge, distributed knowledge, common knowledge, and everybody knowing [8]. BMC [3] consists in translating the model checking problem of an existential CTLK formula (a formula containing only existential modalities) into the problem of satisfiability of a propositional formula. In particular, the BMC algorithm checks for a finite witness among all (possibly infinite) traces of the system satisfying a given existential CTLK formula. While preliminary experimental results seem largely positive [9], unfortunately, the BMC algorithm can only be used efficiently when one wants to check whether an existential CTLK formula is true on a particular system, or whether a universal CTLK formula (one containing only universal modalities) is false[1]. Unfortunately in the applications, it is sometimes the case that one actually does want to check the validity of a universal formula. For example, in a multi-agent security example, one might want to check that a particular fact p (perhaps representing the fact that a particular secret shared key is valid) remains commonly known among the group Γ forever in the future (represented as $AGC_{\Gamma}p$ in CTLK). For the reasons above this sort of formulas are problematic in BMC.

The aim of the present paper is to report preliminary results on a variation, still based on SAT-translation, of the BMC approach in the context of temporal epistemic logic. This other method, called *unbounded model checking* (UMC) has recently been proposed by McMillan [12] in the context of plain temporal logic. To do this and to compare the results obtained under BMC and UMC we also extend the BMC methodology to deal with the existential fragment of the logic CTLK extended by past modalities.

Like any SAT-based method, UMC is based on the translation of the model checking problem into the satisfiability problem of a propositional formula. It differs from BMC in the encoding of the formula to be checked while it still uses the same encoding for the transition relation of the model. UMC exploits

[1]In the BMC method checking the validity of a universal CTLK formula (or the unsatisfiability of an existential one) amounts to checking the whole model thereby negating the main benefits of it.

the characterisation of the basic modalities in Quantified Boolean Formulas (QBF) and the algorithms that translate QBF and fixed point equations over QBF into propositional formulas.

The rest of the paper is structured in the following way. Section 2 introduces interpreted system semantics that provides a semantical basis for MAS and it is used in both methods. The logic $\mathrm{CTL_pK}$ is defined in Section 3. The bounded model checking method is introduced in Section 4. Section 5 discusses unbounded model checking, a variation of bounded model checking. In Section 6 we exemplify the two techniques, by discussing an example from the literature of MAS.

2. Interpreted systems semantics

We assume familiarity with interpreted system semantics [8]. This can be succinctly defined as follows. Assume a set of agents $A = \{1, \dots, n\}$, a set of local states L_i and possible actions Act_i for each agent $i \in A$, and a set L_e and Act_e of local states and actions for the environment. The set of possible global states for the system is defined as $G = L_1 \times \cdots \times L_n \times L_e$, where each element $(l_1, \dots, l_n, l_e)$ of G represents a possible computational state for the whole system. In the following we shall consider the environment component simply as another agent, so we shall consider tuples of n components. Further assume a set of protocols $P_i : L_i \rightarrow 2^{Act_i}$, for $i = 1, \dots, n$, representing the functioning behaviour of every agent, and a protocol $P_e : L_e \rightarrow 2^{Act_e}$ for the environment. We can model the computation taking place in the system by means of a transition function $t : G \times Act \rightarrow G$, where $Act \subseteq Act_1 \times \cdots \times Act_n \times Act_e$ is the set of joint actions. Intuitively, given an initial state ι, the set of protocols, and the transition function, we can build a (possibly infinite) structure that represents all the possible computations of the system. Since in the description of the model checking method we do not consider actions and protocols explicitly, we abstract from these, and use a successor relation to model the temporal evolution. In the example of the final section we show how agents and protocols can be used explicitly.

Definition 2.1. (Models)
Given a set of agents $A = \{1, \dots, n\}$ a temporal epistemic *model* (or simply a model) is a pair $\mathrm{M} = (\mathcal{K}, \mathcal{V})$ with $\mathcal{K} = (G, W, T, \sim_1, \dots, \sim_n, \iota)$, where

- G is the set of the global states for the system (henceforth called simply "states"),
- W is a set of reachable global states from ι, i.e., $W = \{s \in G \mid (\iota, s) \in T^*\}$[2],
- $T \subseteq G \times G$ is a binary (successor) relation on G, such that, $(\forall s \in G)(\exists s' \in G)((s, s') \in T)$,
- $\sim_i \subseteq G \times G$ is an *epistemic accessibility relation* for each agent $i \in A$ defined by $s \sim_i s'$ iff $l_i(s') = l_i(s)$, where the function $l_i : G \rightarrow L_i$ returns the local state of agent i from a global state s, obviously $\sim_i$ is an equivalence relation,
- $\iota \in W$ is the initial state,
- $\mathcal{V} : G \longrightarrow 2^{\mathcal{PV}_K}$ is a valuation function for a set of propositional variables $\mathcal{PV}_K$ such that **true** $\in \mathcal{V}(s)$ for all $s \in G$. $\mathcal{V}$ assigns to each state a set of propositional variables that are assumed to be true at that state.

[2] T^* denotes the reflexive and transitive closure of T.

Note that in the definition above we include both all possible states and the subset of reachable states. The reason for this follows from having past modalities in the language, which are defined over any possible global state so that a simple fixed point semantics for them can be given. Still, note that, if required, it is possible to restrict the range of the past modalities to reachable states only by insisting that the target state is itself reachable from the initial state.

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

Epistemic relations. We extend the concepts of private knowledge and introduce group knowledge in the usual way [8]. Let $\Gamma \subseteq A$. Given the epistemic relations for the agents in Γ, the union of Γ's accessibility relations defines the epistemic relation corresponding to the modality of everybody knows: $\sim^E_\Gamma = \bigcup_{i\in\Gamma} \sim_i$. $\sim^C_\Gamma$ denotes the transitive closure of $\sim^E_\Gamma$, and corresponds to the relation used to interpret the modality of common knowledge. The relation used to interpret the modality of distributed knowledge is given by taking the intersection of the relations corresponding to the agents in Γ, $\sim^D_\Gamma = \bigcap_{i\in\Gamma} \sim_i$. We refer to [8] for an introduction to these concepts.

Computations. A *computation* in M is a possibly infinite sequence of states $\pi = (s_0, s_1, \ldots)$ such that $(s_m, s_{m+1}) \in T$ for each $m \in \mathbb{N}$. Specifically, we assume that $(s_m, s_{m+1}) \in T$ iff $s_{m+1} = t(s_m, act_m)$, i.e., s_{m+1} is the result of applying the transition function t to the global state s_m, and a joint action act_m. All the components of act_m are prescribed by the protocols $P_i, i \in A$, when evaluated on the corresponding local states of s_m. In the following we abstract from the transition function, the actions, and the protocols, and simply use T, but it should be clear that this is uniquely determined by the interpreted system under consideration. In interpreted systems terminology a computation is a part of a run. A k-computation is a computation of length k. For a computation $\pi = (s_0, s_1, \ldots)$, let $\pi(k) = s_k$, and $\pi_k = (s_0, \ldots, s_k)$, for each $k \in \mathbb{N}$. By $\Pi(s)$ we denote the set of all the infinite computations starting at s in M, whereas by $\Pi_k(s)$ the set of all the k-computations starting at s.

3. Computation Tree Logic of Knowledge with Past ($\mathrm{CTL_pK}$)

Interpreted systems are traditionally used to give a semantics to an epistemic language enriched with temporal connectives based on linear time [8]. Here we use CTL by Emerson and Clarke [7] as our basic temporal language and add an epistemic and past component to it. We call the resulting logic Computation Tree Logic of Knowledge with Past ($\mathrm{CTL_pK}$).

Definition 3.1. (Syntax of $\mathrm{CTL_pK}$)
Let $\mathcal{PV}_K$ be a set of propositional variables containing the symbol **true** and let A be a set of agents $\{1, \ldots, n\}$. The set of $\mathrm{CTL_pK}$ formulas $F_{\mathrm{CTL_pK}}$ is defined inductively as follows:

- every member p of $\mathcal{PV}_K$ is a formula,
- if α and β are formulas, then so are $\neg\alpha$, $\alpha \wedge \beta$ and $\alpha \vee \beta$,
- if α and β are formulas, then so are $\mathrm{AX}\alpha$, $\mathrm{AG}\alpha$ and $\mathrm{A}(\alpha \mathrm{U} \beta)$,
- if α is formula, then so are $\mathrm{AY}\alpha$ and $\mathrm{AH}\alpha$,

- if α is formula, then so is $\mathrm{K}_i\alpha$, for $i \in A$,
- if α is formula, then so are $\mathrm{D}_\Gamma\alpha$, $\mathcal{C}_\Gamma\alpha$, and $\mathrm{E}_\Gamma\alpha$, for $\Gamma \subseteq A$.

The remaining basic modalities are defi ned by derivation as follows:

- $\mathrm{EF}\alpha \overset{def}{=} \neg\mathrm{AG}\neg\alpha$, $\mathrm{EP}\alpha \overset{def}{=} \neg\mathrm{AH}\neg\alpha$, $\mathrm{EZ}\alpha \overset{def}{=} \neg\mathrm{AZ}\neg\alpha$, for $\mathrm{Z} \in \{\mathrm{X}, \mathrm{Y}\}$,
- $\overline{\mathrm{K}}_i\alpha \overset{def}{=} \neg\mathrm{K}_i\neg\alpha$, $\overline{\mathrm{D}}_\Gamma\alpha \overset{def}{=} \neg\mathrm{D}_\Gamma\neg\alpha$, $\overline{\mathcal{C}}_\Gamma\alpha \overset{def}{=} \neg\mathcal{C}_\Gamma\neg\alpha$, $\overline{\mathrm{E}}_\Gamma\alpha \overset{def}{=} \neg\mathrm{E}_\Gamma\neg\alpha$.

Moreover, $\alpha \Rightarrow \beta \overset{def}{=} \neg\alpha \vee \beta$, $\alpha \Leftrightarrow \beta \overset{def}{=} (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$, and **false** $\overset{def}{=}$ ¬**true**. We omit the subscript Γ for the epistemic modalities if $\Gamma = A$, i.e., Γ is the set of all the agents. As customary X, G stand for respectively “at the next step”, and “forever in the future”. Y, H are their past counterparts “at the previous step”, and “forever in the past”. The operator U stands for *Until*; the formula $\alpha\mathrm{U}\beta$, expresses the fact that β eventually occurs and that α holds continuously until then. In what follows, we defi ne two fragments of $\mathrm{CTL_pK}$. The universal fragment, called $\mathrm{ACTL_pK}$, and the existential fragment, called $\mathrm{ECTL_pK}$. Both are used to express properties of a MAS.

The logic $\mathrm{ACTL_pK}$ is the restriction of $\mathrm{CTL_pK}$ such that negation can be applied only to elements of $\mathcal{PV}_K$; the defi nition of the language $F_{\mathrm{ACTL_pK}}$ is identical to Defi nition 3.1 except for $\neg p$ replacing $\neg\alpha$ in the second itemised paragraph. The logic $\mathrm{ECTL_pK}$ is the restriction of $\mathrm{CTL_pK}$ such that its language is defi ned as $F_{\mathrm{ECTL_pK}} = \{\neg\varphi \mid \varphi \in F_{\mathrm{ACTL_pK}}\}$. It is easy to see that $\mathrm{ECTL_pK}$ modal formulas can be written as positive Boolean combinations of: $\mathrm{EX}\alpha$, $\mathrm{EY}\alpha$, $\mathrm{E}(\alpha\mathrm{U}\beta)$, $\mathrm{EG}\alpha$, $\mathrm{EP}\alpha$, $\overline{\mathrm{K}}_i\alpha$, $\overline{\mathrm{D}}_\Gamma\alpha$, $\overline{\mathcal{C}}_\Gamma\alpha$, and $\overline{\mathrm{E}}_\Gamma\alpha$. Obviously, all the propositional formulas over $\mathcal{PV}_K$ are in the language of $\mathrm{ECTL_pK}$.

Definition 3.2. (Interpretation of $\mathrm{CTL_pK}$)
Let M be a model, $s \in G$ be a state, π be a computation, and α, β be formulas of $\mathrm{CTL_pK}$. $\mathrm{M}, s \models \alpha$ 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 defi ned inductively as follows:

$$
\begin{array}{llll}
s \models p & \text{iff} & p \in \mathcal{V}(s), & s \models \alpha \vee \beta \quad \text{iff} \quad s \models \alpha \text{ or } s \models \beta, \\
s \models \neg\alpha & \text{iff} & s \not\models \alpha, & s \models \alpha \wedge \beta \quad \text{iff} \quad s \models \alpha \text{ and } s \models \beta, \\
s \models \mathrm{AX}\alpha & \text{iff} & \forall \pi \in \Pi(s)\ \pi(1) \models \alpha, & \\
s \models \mathrm{AG}\alpha & \text{iff} & \forall \pi \in \Pi(s)\ (\forall_{m \geq 0}\ \pi(m) \models \alpha), & \\
s \models \mathrm{A}(\alpha\mathrm{U}\beta) & \text{iff} & \forall \pi \in \Pi(s)\ (\exists_{m \geq 0}\ [\pi(m) \models \beta \text{ and } \forall_{j<m}\ \pi(j) \models \alpha]), & \\
s \models \mathrm{AY}\alpha & \text{iff} & \forall s' \in G(\text{ if } (s', s) \in T, \text{ then } s' \models \alpha), & \\
s \models \mathrm{AH}\alpha & \text{iff} & \forall s' \in G(\text{ if } (s', s) \in T^*, \text{ then } s' \models \alpha), & \\
s \models \mathrm{K}_i\alpha & \text{iff} & \forall s' \in W(\text{ if } s \sim_i s', \text{ then } s' \models \alpha), & \\
s \models \mathrm{D}_\Gamma\alpha & \text{iff} & \forall s' \in W(\text{ if } s \sim_\Gamma^D s', \text{ then } s' \models \alpha), & \\
s \models \mathrm{E}_\Gamma\alpha & \text{iff} & \forall s' \in W(\text{ if } s \sim_\Gamma^E s', \text{ then } s' \models \alpha), & \\
s \models \mathcal{C}_\Gamma\alpha & \text{iff} & \forall s' \in W(\text{ if } s \sim_\Gamma^C s', \text{ then } s' \models \alpha). &
\end{array}
$$

Definition 3.3. (Validity) A $\mathrm{CTL_pK}$ formula φ is valid in $\mathrm{M} = (\mathcal{K}, \mathcal{V})$ (denoted $\mathrm{M} \models \varphi$) iff $\mathrm{M}, \iota \models \varphi$, i.e., φ is true at the initial state of the model M.

Notice that the past component of $\mathrm{CTL_pK}$ does not contain the modality *Since*, which is a past counterpart of modality *Until* denoted by *U*. Extending the logic by *Since* is possible, but complicates the semantics, so this is not discussed in this paper.

In the the next two sections we introduce two methods for verifying symbolically that temporal epistemic formulas hold in a model.

4. Bounded model checking for $\mathrm{CTL_pK}$

In this section we introduce an algorithm for bounded model checking for $\mathrm{ECTL_pK}$. BMC works by translating both the model and the formula to be checked into propositional formulas. The satisfaction of their conjunction is then checked by an efficient SAT-solver. BMC is particularly efficient when the analysis involves looking for faults in protocols whose key properties are expressed as $\mathrm{ACTL_pK}$ formulas. As it will be clear by the end of this section, the algorithm checks increasingly larger but finite and bounded models in an attempt to verify the negation of the universal formula in consideration. To present the algorithm we first need to define satisfaction on bounded models. We shall then present the translations into propositional formulas, and the algorithm itself.

4.1. Bounded semantics for $\mathrm{ECTL_pK}$

In this subsection we give a *bounded semantics* for $\mathrm{CTL_pK}$ in order to define the *bounded model checking problem* for $\mathrm{ECTL_pK}$, and to translate it subsequently into a satisfiability problem. This formalism is an extension to past modalities of the one presented in [13]. The main idea of the bounded semantics consists in using the notion of k-computation to interpret a formula on a finite *fraction* of the model. Moreover, the translation of the existential path quantifier is restricted to finitely many computations[3]; this is known to be sufficient [14, 13]. Below, we define a k-model and the function, which is used to check whether a k-computation is a loop, i.e., represents an infinite computation.

Definition 4.1. ($k-$model)
Let $\mathrm{M} = (\mathcal{K}, \mathcal{V})$ be a model. A $k-$model for M is a structure $\mathrm{M}_k = ((G, W, P_k, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$, where P_k with $k \in \mathbb{N}_+$ is the set of all the k-computations of M, i.e., $P_k = \bigcup_{s \in G} \Pi_k(s)$. The function *loop*: $P_k \rightarrow 2^{\mathbb{N}}$ is defined as: *loop*$(\pi) \;=\; \{l \;\mid\; 0 \leq l \leq k \;\text{ and }\; (\pi(k),\; \pi(l)) \in T\}$.

Satisfaction for the temporal formulas $\mathrm{EG}\alpha$ in the bounded case depends on whether or not the k-computation π defines a loop, i.e., whether *loop*$(\pi) \neq \emptyset$.

Note that the interpretation of the temporal modalities on bounded semantics is defined for $\mathrm{ECTL_pK}$ and is different from the one of Definition 3.2.

Definition 4.2. (Bounded semantics)
Let M_k be a $k-$model and α, β be formulas of $\mathrm{ECTL_pK}$. $\mathrm{M}_k, s \models \alpha$ 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:

[3]The number of the computations depends on k and the formula to be translated.

$$\begin{array}{lll}
s \models p & \text{iff} & p \in \mathcal{V}(s), \\
s \models \neg p & \text{iff} & p \notin \mathcal{V}(s), \\
s \models \alpha \wedge \beta & \text{iff} & s \models \alpha \ \text{ and } \ s \models \beta, \\
s \models \alpha \vee \beta & \text{iff} & s \models \alpha \ \text{ or } \ s \models \beta, \\
s \models \mathrm{EX}\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = s \ \text{ and } \pi(1) \models \alpha\big), \\
s \models \mathrm{EG}\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = s \ \text{ and } \forall_{0 \leq j \leq k} \pi(j) \models \alpha \ \text{ and } \mathit{loop}(\pi) \neq \emptyset\big), \\
s \models \mathrm{E}(\alpha \mathrm{U} \beta) & \text{iff} & \exists \pi \in P_k\big(\pi(0) = s \text{ and } \exists_{0 \leq j \leq k}\big(\pi(j) \models \beta \ \text{ and } \forall_{0 \leq i < j} \pi(i) \models \alpha\big)\big), \\
s \models \mathrm{EY}\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(k) = s \ \text{ and } \pi(k-1) \models \alpha\big), \\
s \models \mathrm{EP}\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(k) = s \ \text{ and } \exists_{0 \leq j \leq k} \pi(j) \models \alpha\big), \\
s \models \overline{\mathrm{K}}_i\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \text{ and } \exists_{0 \leq j \leq k}\big(\pi(j) \models \alpha \ \text{ and } \ s \sim_i \pi(j)\big)\big), \\
s \models \overline{\mathrm{D}}_\Gamma\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \ \text{ and } \ \exists_{0 \leq j \leq k}\big(\pi(j) \models \alpha \ \text{ and } \ s \sim_\Gamma^D \pi(j)\big)\big), \\
s \models \overline{\mathrm{E}}_\Gamma\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \ \text{ and } \ \exists_{0 \leq j \leq k}\big(\pi(j) \models \alpha \ \text{ and } \ s \sim_\Gamma^E \pi(j)\big)\big), \\
s \models \overline{\mathcal{C}}_\Gamma\alpha & \text{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \ \text{ and } \ \exists_{0 \leq j \leq k}\big(\pi(j) \models \alpha \ \text{ and } \ s \sim_\Gamma^C \pi(j)\big)\big).
\end{array}$$

The above extends to past modalities the bounded semantics of [13]. It is easy to notice that the bounded semantics of the past modalities is similar to their forward counterparts up to the inverse of the transition relation. In this setting we can prove that in some circumstances satisfiability in the $|\mathrm{M}|$-bounded semantics is equivalent to the unbounded one.

Theorem 4.1. (Correctness for BMC**)**
Let $\mathrm{M} = ((G, W, T, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$ be a model, φ be an $\mathrm{ECTL_pK}$ formula and $k = |\mathrm{M}|$. Then, $\mathrm{M}, \iota \models \varphi$ iff $\mathrm{M}_k, \iota \models \varphi$.

Proof:
[sketch] 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:

- $\varphi = \mathrm{EX}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{E}(\alpha\mathrm{U}\beta) \mid \overline{\mathrm{K}}_i\alpha \mid \overline{\mathrm{E}}_\Gamma\alpha \mid \overline{\mathrm{E}}_\Gamma\alpha \mid \overline{\mathcal{C}}_\Gamma\alpha$. By induction hypothesis — see [13] page 173.
- For $\varphi = \mathrm{EY}\alpha$, the proof is similar to the case of $\mathrm{EX}\alpha$,
- For $\varphi = \mathrm{EP}\alpha$, the proof is similar to the case of $\mathrm{E}($**true**$\mathrm{U}\alpha)$.

$\square$

Given that we reasoned on a bounded model of size $|\mathrm{M}|$ 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, i.e., k much smaller than $|\mathrm{M}|$ suffices.

4.2. The BMC algorithm for $\mathrm{ECTL_pK}$

Having defined bounded semantics above, we here present the encodings of the formulas and of the model to be checked, that are required by the algorithm. This is an extension of the method of [13]. The main idea is that we can check φ over M_k by checking satisfiability of a propositional formula

$[\mathrm{M},\varphi]_k = [\mathrm{M}^{\varphi,\iota}]_k \wedge [\varphi]_{\mathrm{M}_k}$, 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 $\mathrm{ECTL_pK}$ formula can be done by means of a SAT-checker. Details of this translation are provided below starting from the encoding of the transitions in the interpreted system under consideration.

Let $\mathrm{M} = (\mathcal{K}, \mathcal{V})$ with $\mathcal{K} = (G, W, T, \sim_1, ..., \sim_n, \iota)$. Recall that the set of global states $G = \times_{i=1}^{n} L_i$ is the Cartesian product of the set of local states (here we treat the environment as one of the agents).

We assume $L_i \subseteq \{0,1\}^{n_i}$, where $n_i = \lceil \log_2(|L_i|) \rceil$ and let $n_1 + \ldots + n_n = m$, i.e., every local state is represented by a sequence consisting of 0's and 1's. Moreover, let D_i, for $i = 1, \ldots n$, be a set of the indexes of the bits of the local states of the agent i in the global states, i.e., $D_1 = \{1, \ldots, n_1\}, \ldots, D_n = \{m - n_n + 1, \ldots, m\}$. Next, let $\mathcal{PV}$ be a set of fresh propositional variables disjoint with $\mathcal{PV}_K$ and $\mathrm{F}_{\mathcal{PV}}$ be a set of propositional formulas over $\mathcal{PV}$.

Furthermore, let $w = (w[1], \ldots, w[m])$, where $w[i] \in \mathcal{PV}$ for each $i = 1, \ldots, m$, be a *global state variable*. Global state variables are used for encoding global states. We use elements of G as valuations[4] of global state variables in formulas of $\mathrm{F}_{\mathcal{PV}}$. For example $w[1] \wedge w[2]$ evaluates to $true$ for the valuation $q = (1, \ldots, 1)$, and it evaluates to $false$ for the valuation $q = (0, \ldots, 0)$. 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, defined below.

Definition 4.3. (Function f_k)
Define a function $f_k : F_{\mathrm{ECTL_pK}} \rightarrow \mathbb{N}$ as follows:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}_K$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(Z\alpha) = f_k(\alpha) + 1$, for $Z \in \{\mathrm{EX}, \mathrm{EY}, \mathrm{EP}, \overline{\mathrm{K}}_i, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma\}$,
- $f_k(\overline{\mathcal{C}}_\Gamma) = f_k(\alpha) + k$,
- $f_k(\mathrm{EG}\alpha) = (k+1) \cdot f_k(\alpha) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$.

We refer to [14, 13] for more details. To construct $[\mathrm{M},\varphi]_k$, we first define a propositional formula $[\mathrm{M}^{\varphi,\iota}]_k$ that constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-computations of M_k. For $1 \leq j \leq f_k(\varphi)$, 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 *lit*: $\{0,1\} \times \mathcal{PV} \rightarrow \mathrm{F}_{\mathcal{PV}}$ be a function defined as follows: *lit*$(0,p) = \neg p$ and *lit*$(1,p) = p$, and w, v be two 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 $s = (s_1, \ldots, s_m)$ of the model, i.e., $s_i = 1$ is encoded by $w[i]$, and $s_i = 0$ is encoded by $\neg w[i]$.

[4]We identify 1 with $true$ and 0 with $false$.

- $p(w)$ is a formula over $w[1], \dots, w[m]$, which is true for a valuation $(s_1, \dots, s_m) \in \{0,1\}^m$ of $(w[1], \dots, w[m])$ iff $p \in \mathcal{V}((s_1, \dots, s_m))$, where $p \in \mathcal{PV}_K$.

  This formula encodes the proposition p of $\mathrm{ECTL_pK}$.

- $H(w, v) := \bigwedge_{i=1}^{m} w[i] \Leftrightarrow v[i]$.

  This formula represents logical equivalence between global state encodings, representing the fact that they represent the same state.

- $H_l(w, v) := \bigwedge_{i \in D_l} w[i] \Leftrightarrow v[i]$.

  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.

- $T(w, v)$ is a formula over the propositions $w[1], \dots, w[m]$, $v[1], \dots, v[m]$, which is true for a valuation $(s_1, \dots, s_m)$ of $(w[1], \dots, w[m])$ and a valuation $(s'_1, \dots, s'_m)$ of $(v[1], \dots, v[m])$ iff $((s_1, \dots, s_m), (s'_1, \dots, s'_m)) \in T$.

- $L_{k,j}(l) := T(w_{k,j}, w_{l,j})$,

  This formula encodes a backward loop connecting the k-th state to the l-th state in the symbolic $k-$computation j, for $0 \le l \le k$.

The translation of $[\mathrm{M}^{\varphi,\iota}]_k$, representing the transitions in the k-model is given by the following defi nition.

Definition 4.4. (Unfolding of Transition Relation)
Let $\mathrm{M}_k = ((G, W, P_k, \sim_1, ..., \sim_n, \iota), \mathcal{V})$ be the $k-$model of M, and φ be an $\mathrm{ECTL_pK}$ formula. The propositional formula $[\mathrm{M}^{\varphi,\iota}]_k$ is defi ned as follows:

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

where $w_{0,0}$, and $w_{i,j}$ for $0 \le i \le k$ and $1 \le j \le f_k(\varphi)$ are global state variables.

$[\mathrm{M}^{\varphi,\iota}]_k$ constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-computations in M_k.

The next step of our algorithm is to translate an $\mathrm{ECTL_pK}$ formula φ into a propositional formula.

Definition 4.5. (Translation for BMC)
Let ι be the initial state of the model and $[\varphi]_k^{[m,n]}$ denote the translation of an $\mathrm{ECTL_pK}$ formula φ at $w_{m,n}$ to a propositional formula.

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

Given the translations above, we can now check φ over M_k by checking satisfiability of the propositional formula $[\mathrm{M}^{\varphi,\iota}]_k \wedge [\varphi]_k^{[0,0]}$. The translation presented above can be shown to be correct and complete. The proof is a generalisation of the proof of Theorem 2 of [13] to the past modalities EY and EP. The reader can easily see that the proof for $\mathrm{EY}\varphi$ is similar to the $\mathrm{EX}\varphi$ case, whereas the proof for EP is similar to the $\mathrm{E}(\mathbf{true}\mathrm{U}\varphi)$ case.

We have all ingredients in place to give the algorithm for bounded model checking for $\mathrm{ECTL_pK}$.

Definition 4.6. (BMC algorithm for $\mathrm{ECTL_pK}$)

```
procedure BMC($\varphi, M$), where $\varphi$ is an $\mathrm{ECTL_pK}$ formula
k:==1,
while $k \le |\mathrm{M}|$
    Translate the transition relation of the $k-$computations of $\mathrm{M}_k$
      into a propositional formula $[\mathrm{M}^{\varphi,\iota}]_k$;
    Translate $\varphi$ over $\mathrm{M}_k$ into a propositional formula $[\varphi]_k^{[0,0]}$;
    if $[\mathrm{M},\varphi]_k := [\mathrm{M}^{\varphi,\iota}]_k \wedge [\varphi]_k^{[0,0]}$ is satisfiable return (true,$\mathrm{M}_k$);
    k:== k + 1;
return false
```

The algorithm above has been implemented and experimental results appear very encouraging [9]. The problem of the algorithm above is that its range of applicability is limited to $\mathrm{ECTL_pK}$ formulas. In MAS this is a limitation as universal formulas to be checked do appear in applications. To solve this problem we present the method below.

5. Unbounded model checking for $\mathrm{CTL_pK}$

In this section we present the method of *unbounded model checking* (UMC) for verifying the whole language of $\mathrm{CTL_pK}$ by means of SAT-translation. The method is an extension to temporal epistemic logic of the original paper by McMillan [12] that deals with CTL only.

The method of UMC differs from BMC in the encoding of the formulas, while it shares with BMC the encoding of the states and the transition relation of the model. It exploits the characterisation of the basic modalities in Quantified Boolean Formulas (QBF) and algorithms that translate QBF and fixed point equations over QBF to propositional formulas. To present this we first introduce fixed point characterisations for formulas of $\mathrm{CTL_pK}$.

5.1. Fixed-point characterisation of $\mathrm{CTL_pK}$

In this subsection we show how the set of states satisfying a $\mathrm{CTL_pK}$ formula can be characterised as a fixed point of an appropriate function. We adapt definitions given in [4].

Let $\mathrm{M} = ((G, W, T, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$ be a model. Notice that the set 2^G of all subsets of G forms a lattice under the set inclusion ordering. Each element G' of the lattice can also be thought of as a *predicate* on G, where the predicate is viewed as being true for exactly the states in G'. The least element in the lattice is the empty set, which we also refer to as **false**, and the greatest element in the lattice is the set G, which we sometimes write as **true**. A function τ mapping 2^G to 2^G is called a *predicate transformer*. A set $G' \subseteq G$ is a *fixed point* of a function $\tau : 2^G \rightarrow 2^G$ if $\tau(G') = G'$.

Whenever τ is monotonic, i.e., $P \subseteq Q$ implies $\tau(P) \subseteq \tau(Q)$, it has the least fixed point denoted $\mu Z.\tau(Z)$ and the greatest fixed point denoted $\nu Z.\tau(Z)$. When $\tau(Z)$ is also $\bigcup$-continuous, i.e., $P_1 \subseteq P_2 \subseteq \ldots$ implies $\tau(\bigcup_i P_i) = \bigcup_i \tau(P_i)$ then $\mu Z.\tau(Z) = \bigcup_{i \geq 0} \tau^i(\mathbf{false})$. When $\tau(Z)$ is also $\bigcap$-continuous, i.e., $P_1 \supseteq P_2 \supseteq \ldots$ implies $\tau(\bigcap_i P_i) = \bigcap_i \tau(P_i)$ then $\nu Z.\tau(Z) = \bigcap_{i \geq 0} \tau^i(\mathbf{true})$ (see [15]).

In order to obtain fixed point characterisations of the modal operators, we identify each $\mathrm{CTL_pK}$ formula α with the set $\langle \alpha \rangle_{\mathrm{M}}$ of states in M at which this formula is true, formally $\langle \alpha \rangle_{\mathrm{M}} = \{s \in G \mid \mathrm{M}, s \models \alpha\}$. If M is known from the context we omit the subscript M. Furthermore, we define functions $\mathrm{AX}, \mathrm{AY}, \mathrm{E}_\Gamma$ from 2^G to 2^G as follows:

- $\mathrm{AX}(Z) = \{s \in G \mid \text{for every } s' \in G \text{ if } (s, s') \in T, \text{ then } s' \in Z\}$,
- $\mathrm{AY}(Z) = \{s \in G \mid \text{for every } s' \in G \text{ if } (s', s) \in T, \text{ then } s' \in Z\}$,
- $\mathrm{E}_\Gamma(Z) = \{s \in G \mid \text{for every } s' \in G \text{ if } (\iota, s') \in T^* \text{ and } s \sim^E_\Gamma s', \text{ then } s' \in Z\}$.

Observe that $\langle \mathrm{AX}\alpha \rangle = \mathrm{AX}(\langle \alpha \rangle)$, $\langle \mathrm{AY}\alpha \rangle = \mathrm{AY}(\langle \alpha \rangle)$, $\langle \mathrm{E}_\Gamma\alpha \rangle = \mathrm{E}_\Gamma(\langle \alpha \rangle)$. Then, the following temporal and epistemic operators may be characterised as the least or the greatest fixed point of an appropriate monotonic ($\bigcap$-continuous or $\bigcup$-continuous) predicate transformer.

- $\langle \mathrm{AG}\alpha \rangle = \nu Z.\langle \alpha \rangle \cap \mathrm{AX}(Z)$,
- $\langle \mathrm{A}(\alpha \mathrm{U} \beta) \rangle = \mu Z.\langle \beta \rangle \cup (\langle \alpha \rangle \cap \mathrm{AX}(Z))$,
- $\langle \mathrm{AH}\alpha \rangle = \nu Z.\langle \alpha \rangle \cap \mathrm{AY}(Z)$,

- $\langle \mathcal{C}_\Gamma \alpha \rangle = \nu Z.\mathrm{E}_\Gamma(\langle \alpha \rangle \cap Z)$.

The first three equations are standard (see [6], [4],), whereas the fourth one is defined analogously taking account that $\sim^C_\Gamma$ is the transitive closure of $\sim^E_\Gamma$.

5.2. The UMC method for $\mathrm{CTL_pK}$

We now present the method of UMC for $\mathrm{CTL_pK}$. In order to have a more succinct notation for complex operations on Boolean formulas, we use *Quantified Boolean Formulas* (QBF), an extension of propositional logic by means of quantifiers ranging over propositions. In BNF: $\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \exists p.\alpha \mid \forall p.\alpha$. The semantics of the quantifiers is defined as follows:

- $\exists p.\alpha$ iff $\alpha(p \leftarrow$ **true**$) \vee \alpha(p \leftarrow$ **false**$)$,
- $\forall p.\alpha$ iff $\alpha(p \leftarrow$ **true**$) \wedge \alpha(p \leftarrow$ **false**$)$,

where $\alpha \in \mathrm{QBF}$, $p \in \mathcal{PV}$ and $\alpha(p \leftarrow \psi)$ denotes substitution with the formula ψ of every occurrence of the variable p in formula α. The notation $\forall v.\alpha$, where $v = (v[1], \ldots, v[m])$ is a vector of propositional variables, is used to denote $\forall v[1].\forall v[2] \ldots \forall v[m].\alpha$.

We also need formulas in conjunctive normal forms. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of zero or more clauses where by a *clause* we mean a disjunction of zero or more *literals*, i.e., propositional variables as well as the negations of these.

Our aim is to translate $\mathrm{CTL_pK}$ formulas into propositional formulas. Specifically, for a given $\mathrm{CTL_pK}$ formula β we compute a corresponding propositional formula $[\beta]_{\mathrm{M}}(w)$ over a global state variable w, which encodes those states of the model M that satisfy the formula. Operationally, we work outward from the most nested sub-formulas. In other words, to compute $[O\alpha]_{\mathrm{M}}(w)$, where O is a modality, we work under the assumption of already having computed $[\alpha]_{\mathrm{M}}(w)$. The formula $[\mathrm{AX}\alpha]_{\mathrm{M}}(w)$ is equivalent to QBF formula $\forall v.(T(w,v) \Rightarrow [\alpha]_{\mathrm{M}}(v))$. Similar equivalences we can obtain for formulas $\mathrm{AY}\alpha, \mathrm{K}_i\alpha, \mathrm{D}_\Gamma\alpha, \mathrm{E}_\Gamma\alpha$. Thus, to calculate the actual translations we use either the fixed point or the QBF characterisation of $\mathrm{CTL_pK}$ formulas together with three basic algorithms. The first one, implemented by the procedure *forall* [12], is used for formulas $\mathrm{Z}\alpha$ such that $\mathrm{Z} \in \{\mathrm{AX}, \mathrm{AY}, \mathrm{K}_i, \mathrm{D}_\Gamma, \mathrm{E}_\Gamma\}$. This procedure eliminates the universal quantifier from a QBF formula representing a $\mathrm{CTL_pK}$ formula, and returns the result in a conjunctive normal form. The second algorithm, implemented by the procedure *gfp*$_Z$, is applied to formulas $\mathrm{Z}\alpha$ such that $\mathrm{Z} \in \{\mathrm{AG}, \mathrm{AH}, \mathcal{C}_\Gamma\}$. This procedure computes the greatest fixed point. For formulas of the form $\mathrm{A}(\alpha \mathrm{U} \beta)$ we use the third procedure, called *lfp*$_{AU}$, which computes the least fixed point. In so doing, given a formula β we obtain a propositional formula $[\beta]_{\mathrm{M}}(w)$ such that β is valid in the model M iff the propositional formula $[\beta]_{\mathrm{M}}(w) \wedge I_\iota(w)$ is satisfiable, i.e., $\iota \in \langle \beta \rangle_{\mathrm{M}}$. Below, we formalise the above discussion.

Definition 5.1. (Translation for UMC)
Given a model M and a $\mathrm{CTL_pK}$ formula ϕ, the propositional translation $[\phi]_{\mathrm{M}}(w)$ is inductively defined as follows:

- $[p]_{\mathrm{M}}(w) := \bigvee_{s \in \langle p \rangle} I_s(w)$, for $p \in \mathcal{PV}_{\mathcal{K}}$,
- $[\neg\alpha]_{\mathrm{M}}(w) := \neg[\alpha]_{\mathrm{M}}(w)$,

- $[\alpha \wedge \beta]_{\mathrm{M}}(w) := [\alpha]_{\mathrm{M}}(w) \wedge [\beta]_{\mathrm{M}}(w)$,
- $[\alpha \vee \beta]_{\mathrm{M}}(w) := [\alpha]_{\mathrm{M}}(w) \vee [\beta]_{\mathrm{M}}(w)$,
- $[\mathrm{AX}\alpha]_{\mathrm{M}}(w) := \mathit{forall}\big(v, (T(w,v) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$,
- $[\mathrm{AY}\alpha]_{\mathrm{M}}(w) := \mathit{forall}\big(v, (T(v,w) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$,
- $[\mathrm{K}_i\alpha]_{\mathrm{M}}(w) := \mathit{forall}\big(v, ((H_i(w,v) \wedge \neg\, \mathit{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$,
- $[\mathrm{D}_\Gamma\alpha]_{\mathrm{M}}(w) := \mathit{forall}\big(v, ((\bigwedge_{i\in\Gamma} H_i(w,v) \wedge \neg\, \mathit{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$,
- $[\mathrm{E}_\Gamma\alpha]_{\mathrm{M}}(w) := \mathit{forall}\big(v, ((\bigvee_{i\in\Gamma} H_i(w,v) \wedge \neg\, \mathit{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$,
- $[\mathrm{AG}\alpha]_{\mathrm{M}}(w) := \mathit{gfp}_{AG}([\alpha]_{\mathrm{M}}(w))$,
- $[\mathrm{A}(\alpha \mathrm{U} \beta)]_{\mathrm{M}}(w) := \mathit{lfp}_{AU}([\alpha]_{\mathrm{M}}(w), [\beta]_{\mathrm{M}}(w))$,
- $[\mathrm{AH}\alpha]_{\mathrm{M}}(w) := \mathit{gfp}_{AH}([\alpha]_{\mathrm{M}}(w))$,
- $[\mathcal{C}_\Gamma\alpha]_{\mathrm{M}}(w) := \mathit{gfp}_{C_\Gamma}([\alpha]_{\mathrm{M}}(w))$.

The algorithm *forall*, given a propositional formula α and a set of variables $v[1], ..., v[m]$, constructs a formula χ equivalent to α and eliminates quantified variables on the fly. It is sufficient since χ is in conjunctive normal form. The description of this procedure is given below.

```
procedure forall$(v,\alpha)$, where $v = (v[1],...,v[m])$ and $\alpha$ is a propositional formula
let $\phi = U_{CNF}(\alpha)$, $\chi =$ true, and $A = \emptyset$
repeat
    if $\phi$ contains false, return $\chi$
    else if conflict
        analyse conflict and backtrack
    else if $A_\phi$ is total
        build a blocking clause $c'$
        remove literals of form $v[i]$ or $\neg v[i]$ from $c'$
        add $c'$ to $\phi$ and $\chi$
    else
        choose a literal $l$ such that $l \notin A$ and $\neg l \notin A$ and add $l$ to A
```

Initially the algorithm assumes an empty assignment A, a formula χ to be **true** and ϕ to be a CNF formula, denoted $U_{CNF}(\alpha)$, which is unsatisfiable if and only if α is valid. The construction of $U_{CNF}(\alpha)$ is standard (e.g. see [12]) and we do not give it here. First, the procedure finds a satisfying assignment for ϕ. The search of an appropriate assignment is based on the Davis-Putnam-Logemann-Loveland approach [5] which makes use of two techniques: *Boolean constraint propagation* (BCP) and *conflict-based learning* (CBL). The first builds an assignment A_ϕ which is an extension of the assignment A and is implied by A and ϕ. Next BCP determines the consequence of A_ϕ. The following three cases may occur:

1. A conflict exists, i.e., there exists a clause in ϕ such that all of its literals are false in A_ϕ. So, the assignment A can not be extended to a satisfying one. If a conflict is detected the CBL finds the reason for the conflict and tries to resolve it. Information about the current conflict may be recorded as clauses, which are then added to the formula ϕ without changing its satisfiability. The algorithm then backtracks, i.e., it changes assignment A by withdrawing one of the previous decisions.
2. A conflict does not exist and A_ϕ is total, i.e., the satisfying assignment is obtained. In this case we generate a new clause which is false in the current assignment A_ϕ (i.e., rules out the satisfying assignment) and whose complement characterises a set of assignments falsifying the formula α. This clause is called a *blocking clause*. The construction of this clause is given in [12]. Next the blocking clause is deprived of the variables either of the form $v[i]$ or the negation of these and then what remains is added to the formulas ϕ and χ and the algorithm again tries to find a satisfying assignment for ϕ.
3. The first two cases do not apply. Then, the procedure makes a new assignment A by giving a value to a selected variable.
On termination, when ϕ becomes unsatisfiable, χ is a conjunction of the blocking clauses and precisely characterises $\forall v.\alpha$.

Theorem 5.1. Let α be a propositional formula and $v = (v[1], \ldots, v[m])$ be a vector of propositions, then the QBF formula $\forall v.\alpha$ is logically equivalent to the CNF formula *forall*(v, α).

The proof of the above theorem follows from the correctness of *forall* algorithm (see [12]).
The algorithms *gfp* and *lfp* are based on the standard procedures computing fixed points.

```
procedure $gfp_{AG}([\alpha]_{\mathrm{M}}(w))$, where $\alpha$ is an $\mathrm{CTL_pK}$ formula
let $Q(w) = [\mathbf{true}]_{\mathrm{M}}(w)$, $Z(w) = [\alpha]_{\mathrm{M}}(w)$
while $\neg(Q(w) \Rightarrow Z(w))$ is satisfiable
    let $Q(w) = Z(w)$,
    let $Z(w) = forall(v, (T(w,v) \Rightarrow Z(v))) \wedge [\alpha]_{\mathrm{M}}(w)$
return $Q(w)$
```

The procedure gfp_{AH} is obtained by replacing in the above $Z(w) = forall(v, (T(w,v) \Rightarrow Z(v))) \wedge [\alpha]_{\mathrm{M}}(w)$ with $Z(w) = forall(v, (T(v,w) \Rightarrow Z(v))) \wedge [\alpha]_{\mathrm{M}}(w)$. Similarly, the procedure gfp_{C_Γ} is obtained by replacing $Z(w) = [\alpha]_{\mathrm{M}}(w)$ with $Z(w) = forall\big(v, ((\bigvee_{i\in\Gamma} H_i(w,v) \wedge \neg gfp_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathrm{M}}(v))\big)$ and $Z(w) = forall(v, (T(w,v) \Rightarrow Z(v))) \wedge [\alpha]_{\mathrm{M}}(w)$ with $Z(w) = forall(v, ((\bigvee_{i\in\Gamma} H_i(w,v) \wedge \neg gfp_{AH}(\neg I_\iota(v))) \Rightarrow (Z(v) \wedge [\alpha]_{\mathrm{M}}(v))))$.

```
procedure $lfp_{AU}([\alpha]_{\mathrm{M}}(w), [\beta]_{\mathrm{M}}(w))$,
where $\alpha, \beta$ are $\mathrm{CTL_pK}$ formulas
let $Q(w) = [\mathbf{false}]_{\mathrm{M}}(w)$, $Z(w) = [\beta]_{\mathrm{M}}(w)$
while $\neg(Z(w) \Rightarrow Q(w))$ is satisfiable
    let $Q(w) = Q(w) \vee Z(w)$,
    let $Z(w) = forall(v, (T(w,v) \Rightarrow Q(v))) \wedge [\alpha]_{\mathrm{M}}(w)$
return $Q(w)$
```

Theorem 5.2. (Correctness of UMC **for** $\mathrm{CTL_pK}$**)**
Let M be a model and φ be a $\mathrm{CTL_pK}$ formula. Then, $\mathrm{M} \models \varphi$ iff $[\varphi]_{\mathrm{M}}(w) \wedge I_\iota(w)$ is satisfiable.

Proof:
[Sketch]

By induction on the length of φ. The theorem follows directly for the propositional variables. Next, assume that the hypothesis holds for all the proper sub-formulas of φ. If φ is equal to either $\neg\alpha$, $\alpha \wedge \beta$, or $\alpha \vee \beta$, then it is easy to check that the theorem holds. Consider φ to be of the following forms:

- $\varphi = \mathrm{K}_i\alpha$. Then, $M, s \models \mathrm{K}_i\alpha$ iff for every state $s' \in G$ if $(\iota, s') \in T^*$ and $s \sim_i s'$, then $\mathrm{M}, s' \models \alpha$. Intuitively, s satisfies $\mathrm{K}_i\alpha$ iff any state s' which is reachable from the initial state and is in epistemic relation $\sim_i$ with the state s, satisfies α. Assume two global variables w and v representing states s and s' respectively. Then, based on the inductive assumption that $\mathrm{M}, s' \models \alpha$ iff $[\alpha]_{\mathrm{M}}(v)$ is true for the valuation $(s'_1, \ldots, s'_m)$ of $(v[1], \ldots, v[m])$ and s' is reachable from the initial state iff the formula $\neg \mathit{gfp}_{AH}(\neg I_\iota(v))$ is true for the valuation $(s'_1, \ldots, s'_m)$ of $(v[1], \ldots, v[m])$, we obtain that s satisfies $\mathrm{K}_i\alpha$ iff the following QBF formula $\forall v.(\neg \mathit{gfp}_{AH}(\neg I_\iota(v)) \wedge H_i(w, v) \Rightarrow [\alpha]_{\mathrm{M}}(v))$ is true for the valuation $(s_1, \ldots, s_m)$ of $(w[1], \ldots, w[m])$. So, $M \models \mathrm{K}_i\alpha$ iff $M, \iota \models \mathrm{K}_i\alpha$ iff $\forall v.(\neg \mathit{gfp}_{AH}(\neg I_\iota(v)) \wedge H_i(w, v) \Rightarrow [\alpha]_{\mathrm{M}}(v))$ is true for the valuation $(\iota_1, \ldots, \iota_m)$ of $(w[1], \ldots, w[m])$ iff the propositional formula $\mathit{forall}(v, \neg \mathit{gfp}_{AH}(\neg I_\iota(v)) \wedge H_i(w, v) \Rightarrow [\alpha]_{\mathrm{M}}(v))$ is true for the valuation $(\iota_1, \ldots, \iota_m)$ of $(w[1], \ldots, w[m])$. The last equivalence follows from Theorem 5.1. Consequently $M, \iota \models \mathrm{K}_i\alpha$ iff $[\mathrm{K}_i\alpha]_{\mathrm{M}}(w)$ is true for the valuation $(\iota_1, \ldots, \iota_m)$ of $(w[1], \ldots, w[m])$ iff $[\mathrm{K}_i\alpha]_{\mathrm{M}}(w) \wedge I_\iota(w)$ is satisfiable.

- $\varphi = \mathrm{AX}\alpha \mid \mathrm{AY}\alpha \mid \mathrm{E}_\Gamma\alpha \mid \mathrm{D}_\Gamma\alpha$. The above formulas can be characterised by QBF formulas, so the proof is analogous to the former case.

- $\varphi = \mathrm{AG}\alpha \mid \mathrm{AH}\alpha \mid \mathcal{C}_\Gamma\alpha \mid \mathrm{A}(\alpha \mathrm{U} \beta)$. The proof is based on the fixed point characterisations of the formulas and correctness of the procedures computing fixed points.

$\square$

Next, we give an algorithm for unbounded model checking of $\mathrm{CTL_pK}$.

Definition 5.2. (UMC algorithm for $\mathrm{CTL_pK}$)

```
**procedure** *UMC* $(\varphi, M, w)$, where $\varphi$ be a $\mathrm{CTL_pK}$ formula, an $w$ be a state variable
Compute the propositional formula $[\varphi]_{\mathrm{M}}(w)$;
Compute the propositional formula $I_\iota(w)$;
if $[\varphi]_{\mathrm{M}}(w) \wedge I_\iota(w)$ is satisfiable return **true**
else return **false**
```

6. Example of Train, Gate and Controller

In this section we exemplify the procedure above by discussing the scenario of the train controller system (adapted from [17]). The system consists of three agents: two trains (agents 1 and 3), and a controller (agent 2). The trains, one Eastbound, the other Westbound, run on a circular track. At one point, both tracks need to go through a narrow tunnel. There is no room for both trains to be in the tunnel at the same

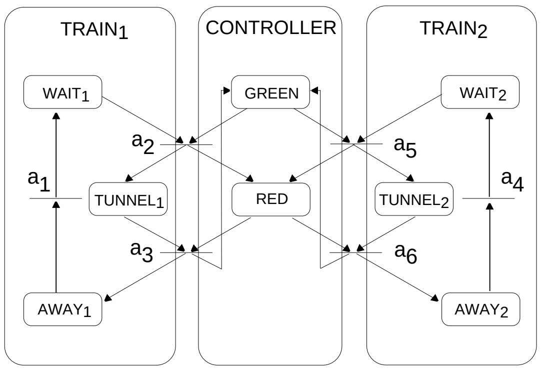

Figure 1. The local transition structures for the two trains and the controller

time, so the trains must avoid this to happen. There are traffi c lights on both sides of the tunnel, which can either be red or green. Both trains are equipped with a signaller, that they use to send a signal when they approach the tunnel. The controller can receive signals from both trains, and controls the colour of the traffi c lights. The task of the controller is to ensure that the trains are never both in the tunnel at the same time. The trains follow the traffi c lights signals diligently, i.e., they stop on red.

We can model the example above with an interpreted system as follows. The local states for the agents are:

- $L_{train_1} = \{away_1, wait_1, tunnel_1\}$,
- $L_{controller} = \{red, green\}$,
- $L_{train_2} = \{away_2, wait_2, tunnel_2\}$.

The set of possible global states is defi ned as $G = L_{train_1} \times L_{controller} \times L_{train_2}$.

Let $\iota = (away_1, green, away_2)$ be the initial state. We assume that the local states are numbered in the following way: $away_1 := 1$, $wait_1 := 2$, $tunnel_1 := 3$, $red; = 4$, $green := 5$, $away_2 := 6$, $wait_2 := 7$, $tunnel_2 := 8$ and the agents are numbered as follows: $train_1 := 1$, $controller := 2$, $train_2 := 3$. Thus we assume a set of agents A to be the set $\{1, 2, 3\}$.

Let $Act = \{a_1, ..., a_6\}$ be a set of joint actions. For $a \in Act$ we defi ne the preconditions $pre(a)$, postconditions $post(a)$, and the sets $agent(a)$ containing the numbers of the agents that may change local states by executing a.

- $pre(a_1) = \{1\}, post(a_1) = \{2\}$, $agent(a_1) = \{1\}$,
- $pre(a_2) = \{2, 5\}, post(a_2) = \{3, 4\}$, $agent(a_2) = \{1, 2\}$,
- $pre(a_3) = \{3, 4\}, post(a_3) = \{1, 5\}$, $agent(a_3) = \{1, 2\}$,
- $pre(a_4) = \{6\}, post(a_4) = \{7\}$, $agent(a_4) = \{3\}$,
- $pre(a_5) = \{5, 7\}, post(a_5) = \{4, 8\}$, $agent(a_5) = \{2, 3\}$,

- $pre(a_6) = \{4,8\}, post(a_6) = \{5,6\}$, $agent(a_6) = \{2,3\}$.

In our formulas we use the following two propositional variables in_tunnel_1 and in_tunnel_2 such that $in\_tunnel_1 \in \mathcal{V}(s)$ iff $l_{train_1}(s) = tunnel_1$, $in\_tunnel_2 \in \mathcal{V}(s)$ iff $l_{train_2}(s) = tunnel_2$, for $s \in G$.

We now encode the local states in binary form in order to use them in the model checking technique. Given that agent $train_1$ can be in 3 different local states we shall need 2 bits to encode its state; in particular we shall take: $(0,0) = away_1$, $(1,0) = wait_1$, $(0,1) = tunnel_1$. Similarly for the agent $train_2$: $(0,0) = away_2$, $(1,0) = wait_2$, $(0,1) = tunnel_2$. The modelling of the local states of the controller requires only one bit: $(0) = green$, $(1) = red$. In view of this a global state is modelled by 5 bits. For instance the initial state $\iota = (away_1, green, away_2)$ is represented as a tuple of 5 0's. Notice that two fi rst bits of a global state encode local states of agent 1, the third bit encodes local states of agent 2, and two remaining bits encode local states of agent 3. Hence, the sets of labels of bits represented local states are the following: $D_1 = \{1,2\}$, $D_2 = \{3\}$, $D_3 = \{4,5\}$.

Let $w = (w[1], ..., w[5])$, $v = (v[1], ..., v[5])$ be two global state variables. We defi ne the following propositional formulas over w and v:

- $I_\iota(w) := \bigwedge_{j \in D_1 \cup D_2 \cup D_3} \neg w[j]$,

  this formula encodes the initial state,

- $H_l(w,v) := \bigwedge_{j \in D_l} w[j] \Leftrightarrow v[j]$, for $l = 1,2,3$,

  the formula $H_l(w,v)$ for $l = 1,2,3$, represents logical equivalence between local states of agent l at two global states represented by variables w and v,

- $p_1(w) := \neg w[1] \wedge \neg w[2]$, $p_2(w) := w[1] \wedge \neg w[2]$, $p_3(w) := \neg w[1] \wedge w[2]$, $p_4(w) := w[3]$, $p_5(w) := \neg w[3]$, $p_6(w) := \neg w[4] \wedge \neg w[5]$, $p_7(w) := w[4] \wedge \neg w[5]$, $p_8(w) := \neg w[4] \wedge w[5]$,

  the formula $p_j(w)$ for $j = 1, \ldots, 8$ encodes the local state j.

For $a \in Act$ let $B_a := \bigcup_{i \in A \setminus agent(a)} D_i$ be the set of the labels of the bits that are not changed by the action a, then

- $T(w,v) := \bigvee_{a \in Act} \big(\bigwedge_{j \in pre(a)} p_j(w) \wedge \bigwedge_{j \in post(a)} p_j(v) \wedge \bigwedge_{j \in B_a} (w[j] \Leftrightarrow v[j]) \big) \vee$ $(\bigwedge_{a \in Act} \bigvee_{j \in pre(a)} (\neg p_j(w)) \wedge \bigwedge_{j \in D_1 \cup D_2 \cup D_3} (w[j] \Leftrightarrow v[j]))$.

  Intuitively, $T(w,v)$ encodes the set of all couples of global states s and s' represented by variables w and v respectively, such that s' is reachable from s, i.e., either there exists a joint action which is available at s and s' is the result of execution a at s or there is not such action and s' equals s. Notice that the above formula is composed of two parts. The fi rst one encodes the transition relation of the system whereas the second one adds self-loops to all the states without successors. This is necessary in order to satisfy the assumption that T is total.

Consider the following two formulas:

- $\alpha_1 \equiv \mathrm{AG}(in\_tunnel_1 \Rightarrow \mathrm{K}_{train_1}(\neg in\_tunnel_2))$ and

- $\alpha_2 \equiv \mathrm{AG}(\neg in\_tunnel_1 \Rightarrow (\neg \mathrm{K}_{train_1} in\_tunnel_2 \wedge \neg \mathrm{K}_{train_1}(\neg in\_tunnel_2)))$.

The first formula expresses that when the agent $train_1$ is in the tunnel, it knows the agent $train_2$ is not in the tunnel. The second formula expresses that when the agent $train_1$ is away from the tunnel, it does not know whether or not the agent $train_2$ is in the tunnel. The translations of α_1 and α_2 into propositional formulas are the following[5]:

- $[\alpha_1]_M(w) = true$,
- $[\alpha_2]_M(w) = \neg w[1] \vee \neg w[2]$.

Since the conjunctions $[\alpha_1]_M(w) \wedge I_\iota(w)$ and $[\alpha_2]_M(w) \wedge I_\iota(w)$ are satisfiable both formulas are valid in the model.

Notice that α_1 is an $\mathrm{ACTL_pK}$ formula, so it can be verified using both the methods BMC and UMC, whereas α_2 can be only verified using UMC.

7. Conclusions

Formal methods in multi-agent systems have traditionally been associated with specifications. Recently, *verification* of multi-agent systems has become an active area of research. In this paper we have reported the extension to epistemic notions of two SAT-based techniques, BMC, and UMC, for verification of reactive systems.

The framework described in the previous sections allows us to verify the temporal epistemic properties of MAS. In principle, by means of BMC and UMC on $\mathrm{CTL_pK}$ we can check formulas representing:

- Private and group knowledge of a MAS about a changing world,
- Temporal evolution of knowledge in a MAS,
- Any combination of the above.

Each of the two methods has advantages and disadvantages.

BMC is limited to the verification of properties expressed by existential $\mathrm{CTL_pK}$ formulas (or the falsification of properties expressed by universal ones). Moreover the method cannot be applied for checking formulas with "mixed formulas" like, for example, $\mathrm{AG}\overline{\mathrm{K}}_i\alpha$ or $\mathcal{C}_\Gamma\mathrm{EF}\alpha$. On the positive side the verification step can be performed on a part of the model only, which may have critical influence on the efficiency and feasibility of the approach.

UMC has no restriction on the syntax of formulas to be verified. However, as the encoding of the full transition relation is used for computing fixed points, the feasibility of this depends on the number of iterations that are required.

For verifying existential $\mathrm{CTL_pK}$ properties BMC and UMC can be viewed as two complementary methods, which should be run in parallel in order to increase the probability of a successful verification.

Our initial investigation looks promising both in terms of possible experimental results, and in terms of possible extension to other modalities dealing with different informational, motivational, or normative aspects of multi-agent systems. This is left for further work.

[5]For simplicity, we show only formulas equivalent to propositional formulas obtained after the execution of the algorithms gfp and $forall$.

References

[1] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.

[2] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking agentspeak. In J. S. Rosenschein, T. Sandholm, W. Michael, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent systems (AAMAS-03)*, pages 409–416. ACM Press, 2003.

[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. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[5] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7):394–397, 1962.

[6] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. of the 7th Int. Colloquium on Automata, Languages and Programming (ICALP'80)*, volume 85 of *LNCS*, pages 169–181. Springer-Verlag, 1980.

[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] A. Lomuscio, T. Łasica, and W. Penczek. Bounded model checking for interpreted systems: Preliminary experimental results. In *Proc. of the 2nd NASA Workshop on Formal Approaches to Agent-Based Systems (FAABS'02)*, volume 2699 of *LNAI*, pages 115–125. Springer-Verlag, 2003.

[10] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[11] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[12] K. L. McMillan. Applying SAT methods in unbounded symbolic model checking. In *Proc. of the 14th Int. Conf. on Computer Aided Verifi cation (CAV'02)*, volume 2404 of *LNCS*, pages 250–264. Springer-Verlag, 2002.

[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] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacifi c Journal of Mathematics*, 5:285–309, 1955.

[16] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.

[17] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proc. of the 1st Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, volume III, pages 1167–1174. ACM, July 2002.

[18] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of the 19th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.

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