Wojciech Penczek and Alessio Lomuscio

Bounded Model Checking for Interpreted Systems*

Nr 946

Warsaw, July 2002

Abstract

We present an extension of a bounded model checking algorithm to deal with temporal epistemic formulas. We use the algorithm to solve a variant of the bit transmission problem.
Keywords: Interpreted systems, temporal logic, logic of knowledge, bounded model checking, translation to SAT, system verification.

Streszczenie

Ograniczona weryfikacja modelowa dla interpretowanych systemów. W pracy przedstawiamy rozszerzenie algorytmu ograniczonej weryfikacji modelowej do temporalnych formul epistemicznych. Algorytm jest zastosowany do weryfikacji pewnego wariantu problemu transmisji bitów.
Słowa kluczowe: Interpretowane systemy, logika temporalna, logika wiedzy, ograniczona weryfikacja modelowa, translacja do SAT, weryfikacja systemów.

* Partly supported by the State Committee for Scientific Research under the grant No. 7T11C 00620, and by the EU Framework V research project ALFEBIITE (IST-1999-10298).

1 Introduction

The field of multiagent systems (MAS) theories is traditionally concerned with the formal representation of the mental attitudes of autonomous entities, or *agents*, in a distributed system. For this task several modal logics have been developed in the past 20 years, the most studied being logics for knowledge, beliefs, desires, goals, and intentions.

These logics are seen as *specifications* of particular classes of MAS systems. Their aim is to offer a description of the *macroscopic* mental properties (such as knowledge, beliefs, etc.) that a MAS should exhibit in a specific class of scenarios. Sometimes, *interaction properties* are studied. For example, in an epistemic and doxastic model of agency it often makes sense to impose that knowledge is true belief. This leads to a logic with two families of modalities, $\{K_i\}_{i \in A}, \{B_i\}_{i \in A}$, where B_i is a KD45-modality, K_i is an S5-modality, and the interaction axiom $K_i p \rightarrow B_i p$ expresses the intended interplay between the two informational operators. A considerable number of these formal studies are available in the literature and temporal extensions of these (i.e., modal combinations of CTL [Eme90] or LTL [MP92] with the modalities for the mental attitudes) have appeared recently. The typical technical contribution of this line of work is to explore the metalogical properties of these logics, e.g., completeness, decidability, and computational complexity.

While these investigations are conceptually valuable, a deep computational analysis of the system is often missing. Typically, the semantics given for these logics is a plain Kripke models semantics which abstracts from how the *actual computation* proceeds in the MAS. Attempts have been made to solve this problem by giving a clear computational model on which to define informational modalities. The framework of *interpreted systems* [FHMV95] is one of such formalisms. There, the notions of actions, protocols, and transitions are given a prominent role, and epistemic modalities are defined, not on abstract possible worlds, but on the sets of runs that actually encode all the possible computations of the system. This has provided a formal basis for an in-depth analysis of the epistemic properties that arise in a MAS when the systems enjoys particular properties such as synchronicity, asynchronicity, perfect recall, no learning, broadcasting, etc.

The advantage of having a computationally grounded semantics such as the one of interpreted systems is that it allows for the possibility of exploring the issue of *verifying* MAS. In particular, interpreted systems are very promising because they are built upon standard CTL semantics, allowing for possible integration with model checking tools [CGP99]. Indeed, some investigations along these lines have been carried out already. In particular, [vdHW02a, vdHW02b] analyze respectively application of SPIN and MOCHA to model checking of LTL and ATL extended by epistemic modalities, whereas [MS99] studies the complexity of the model checking problem for essentially infinite state systems of knowledge and time.

Model checking provides for a promising set of techniques for hardware and software verification, but it suffers from what is known as the state explosion problem. Essentially, in this formalism checking that a property follows from a specification amounts to checking whether or not a modal formula is valid on a model representing all possible computations of the system. Encoding such a model is not problematic for small tailored examples, but it quickly becomes unfeasible as the number of states increases. Moreover, and perhaps most importantly, a full generation of the model is simply not possible if the specification actually generates an infinite computational model. One of the most promising solutions to these two problems is bounded model checking (BMC) [BCCZ99, CBRZ01]. In this line of work, a methodology is developed to explore only the part of the model that is sufficient to validate the particular formula that needs to be checked. Then, the model checking problem over the part of the model is translated into a test of propositional satisfiability, for which refined tools already exist [MMZZM01].

The aim of this abstract is to report on recent progress on the application of bounded model checking to verifying not just temporal, but also epistemic properties of a MAS. Our approach is novel in the way it combines CTL with knowledge modalities in a fully automatic fashion. Given space consideration of the abstract, we present here only the main ideas of the method and sketch the applicability of it to a particular example. Full details of both will be given in the full paper.

The rest of the paper is organized as follows. Section 2 introduces interpreted system semantics. The logic **CTLK** is defined in Section 3. In the next section, bounded semantics of **CTLK** is presented. Section 5 describes the BMC algorithm for **CTLK**. The two final sections contain analysis of the bit transmission problem and conclusions.

2 Interpreted systems semantics

We assume familiarity with interpreted system semantics [FHMV95]. This can be succinctly defined as follows. Assume a set of agents $A = \{1, \ldots, n\}$, 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 global states for the system is defined as $G \subseteq L_1 \times \ldots \times L_n \times L_e$, where each element $(l_1, \ldots, l_n, l_e)$ of G represents a computational state for the whole system. Further assume a set of protocols $P_i : L_i \rightarrow Act_i$, for $i = 1, \ldots, n$, representing the the functioning behaviour of every agent, and a function $P_e : L_e \rightarrow Act_e$ for the environment[1]. We can model the computation taking place in the system by means of a transition function $t : G \times Act \rightarrow G$, where $Act \subseteq Act_1 \times \ldots \times Act_n \times Act_E$ is the set of joint actions. Intuitively, given an initial state s_0, the sets of protocols, and the transition function, we can build a (possibly infinite) structure that represents all the possible computations of the system. Many representations can be given to this structure; since in this paper we are only concerned with temporal epistemic properties, we shall find the following to be a useful one.

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

- *W is a finite[2] set of reachable global states for the system (henceforth called simply "states"),*
- *$T \subseteq W \times W$ is a total[3] binary (successor) relation on W,*
- *$\sim_i \subseteq W \times W$ $(i \in A)$ is an* epistemic accessibility relation *for each agent $i \in A$ defined by $w \sim_i w'$ iff $l_i(w') = l_i(w)$, where the function $l_i : W \rightarrow L_i$ returns the local state of agent i from a global state w. Obviously $\sim_i$ is an equivalence relation.*
- *$\iota \in W$ is the initial state,*
- *$\mathcal{V} : W \longrightarrow 2^{\mathcal{PV}}$ is a valuation function for a set of propositional variables $\mathcal{PV}$ such that* **true** $\in \mathcal{V}(s)$ *for all $s \in W$. $\mathcal{V}$ assigns to each state a set of propositional variables that are assumed to be true at that state.*

By $|M|$ 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. Let $\Gamma \subseteq N$. The union of Γ's accessibility relations is defined as $\sim^E_\Gamma = \bigcup_{i \in \Gamma} \sim_i$. By $\sim^C_\Gamma$ we denote the transitive closure of $\sim^E_\Gamma$, whereas $\sim^D_\Gamma = \bigcap_{i \in \Gamma} \sim_i$. The above relations are later used to give semantics to the "common knowledge", "everyone knows", and "distributed knowledge" modalities of the logic [FHMV95].

Computations paths. A *computation* in M is a possibly infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $(s_i, s_{i+1}) \in T$ for each $i \in \mathbb{N}$[4]. A k-computation is a sequence 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 a set of all the infinite computations starting at s in M, whereas by $\Pi_k(s)$ a set of all the k-computations starting at s.

3 Computation Tree Logic of Knowledge (CTLK)

Interpreted systems are traditionally used to give a semantics to an epistemic language enriched with temporal connectives based on linear time [FHMV95]. Here we use CTL by Emerson and Clarke [EC82] as our basic temporal language and add an epistemic component to it. We call the resulting logic Computation Tree Logic of Knowledge (**CTLK**).

Definition 2 (Syntax of CTLK) *Let $\mathcal{PV}$ be a set of propositional variables containing the symbol* **true**. *The set of* **CTLK** *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* $\mathrm{E}(\alpha \mathrm{U} \beta)$,
- *if α is formula, then so are $\overline{\mathrm{K}}_i\alpha$, for $i \in A$,*

[1] No technical difficulty arises by assuming nondeterminism.

[2] For simplicity in this paper we use finite models but the formalism presented here can be extended to infinite models with no difficulty by considering different sets of languages for the local states depending on the size of the bounded model under consideration. This will be clear later on.

[3] By "total" we meant that given any w there exists a w' such that wRw' — this condition is also called "seriality" elsewhere in modal logic.

[4] Intuitively $(s_i, s_{i+1}) \in T$ iff $s_{i+1} = t(s_i, act_i)$, i.e., s_{i+1} is the result of applying the transition function t to a global state s_i and a joint action act_i in which all the components of act_i are prescribed by the corresponding protocol P_i at s_i. In the following we abstract from the transition function, the actions, and the protocols, and simply use T. In interpreted systems terminology a computation is a part of a run.

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

The basic modalities are defined by derivation as follows:

- $\mathrm{AF}\alpha \stackrel{def}{=} \neg\mathrm{EG}\neg\alpha$, $\mathrm{A}(\alpha\mathrm{R}\beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\alpha\mathrm{U}\neg\beta)$, $\mathrm{AX}\alpha \stackrel{def}{=} \neg\mathrm{EX}\neg\alpha$,

- $\mathrm{K}_i\alpha \stackrel{def}{=} \neg\overline{\mathrm{K}}_i\neg\alpha$, $\mathrm{D}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathrm{D}}_\Gamma\neg\alpha$, $\mathcal{C}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathcal{C}}_\Gamma\neg\alpha$, $\mathrm{E}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathrm{E}}_\Gamma\neg\alpha$.

Moreover, $\alpha \rightarrow \beta$ is defined as $\neg\alpha \vee \beta$.

The logic **ECTLK** is the restriction of **CTLK** such that the negation can be applied only to elements of $\mathcal{PV}$ — the definition of **ECTLK** is identical to Definition 2 except for $\neg p$ replacing $\neg\alpha$. The logic **ACTLK** is the restriction of **CTLK** such that its language is defined as $\{\neg\varphi \mid \varphi \in \mathbf{ECTLK}\}$. It is easy to see that **ACTLK** consists of the formulas of the form: $\mathrm{AX}\alpha$, $\mathrm{A}(\alpha\mathrm{R}\beta)$, $\mathrm{AF}\alpha$, $\mathrm{K}_i\alpha$, $\mathrm{D}_\Gamma\alpha$, $\mathcal{C}_\Gamma\alpha$, and $\mathrm{E}_\Gamma\alpha$.

Definition 3 (Interpretation of CTLK) *Let* M *be a model,* s *be a state,* π *be a computation, and* α, β *be formulas of* **CTLK**. $\mathrm{M}, s \models \alpha$ *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 defined inductively as follows:*

$s \models p$	*iff*	$p \in \mathcal{V}(s)$,
$s \models \neg\alpha$	*iff*	$s \not\models \alpha$,
$s \models \alpha \vee \beta$	*iff*	$s \models \alpha$ *or* $s \models \beta$,
$s \models \alpha \wedge \beta$	*iff*	$s \models \alpha$ *and* $s \models \beta$,
$s \models \mathrm{EX}\alpha$	*iff*	$\exists\pi \in \Pi(s)\ \pi(1) \models \alpha$,
$s \models \mathrm{EG}\alpha$	*iff*	$\exists\pi \in \Pi(s)\ \forall_{m\geq 0}\ \pi(m) \models \alpha$,
$s \models \mathrm{E}(\alpha\mathrm{U}\beta)$	*iff*	$\exists\pi \in \Pi(s)\ (\exists_{m\geq 0}\ [\pi(m) \models \beta$ *and* $\forall_{j<m}\ \pi(j) \models \alpha])$,
$s \models \overline{\mathrm{K}}_i\alpha$	*iff*	$\exists s' \in W\ (s \sim_i s'$ *and* $s' \models \alpha)$,
$s \models \overline{\mathrm{D}}\alpha$	*iff*	$\exists s' \in W\ (s \sim^D_\Gamma s'$ *and* $s' \models \alpha)$,
$s \models \overline{\mathrm{E}}_\Gamma\alpha$	*iff*	$\exists s' \in W\ (s \sim^E_\Gamma s'$ *and* $s' \models \alpha)$,
$s \models \overline{\mathcal{C}}_\Gamma\alpha$	*iff*	$\exists s' \in W\ (s \sim^C_\Gamma s'$ *and* $s' \models \alpha)$.

Definition 4 (Validity) *A* **CTLK** *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.

4 Bounded Semantics of CTLK

In this section we give a *bounded semantics* of **CTLK** in order to define the *bounded model checking problem* for **ECTLK** and to translate it into the satisfiability problem. This is an extension of [PWZ02].

Definition 5 (k−model) *Let* $M = (\mathcal{K}, \mathcal{V})$ *be a model and* $k \in \mathbb{N}_+$. *The* k−*model for* M *is a structure* $M_k = ((W, P_k, \iota), \mathcal{V})$, *where* P_k *is the set of all the* k-*computations of* M.

The bounded semantics is defined over a k−model M_k. The definition of the bounded semantics of the temporal operators depends on whether a considered k−computation of M_k constitute a loop or not. In order to distinguish which of the k−computations constitute loops we define the following auxiliary function.

Definition 6 *Let* $M = ((W, T, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$ *be a model,* $M_k = ((W, P_k, \iota), \mathcal{V})$ *be a* k−*model for* M *and* $\pi \in P_k$ *be a* k−*computation. A function* $loop : P_k \rightarrow 2^{\mathbb{N}}$ *is defined as:*

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

It is easy to notice that π is a loop if $loop(\pi) \neq \emptyset$.

Definition 7 (Bounded semantics) *Let M_k be a $k-$model and α, β be* **ECTLK** *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 implicitly understood. The relation $\models$ is defined inductively as follows:*

$$
\begin{array}{lll}
s \models p & \textit{iff} & p \in \mathcal{V}(s), \\
s \models \neg p & \textit{iff} & p \notin \mathcal{V}(s), \\
s \models \alpha \vee \beta & \textit{iff} & s \models \alpha \ \ or \ \ s \models \beta, \\
s \models \alpha \wedge \beta & \textit{iff} & s \models \alpha \ \ and \ \ s \models \beta, \\
s \models \mathrm{EX}\alpha & \textit{iff} & \exists \pi \in P_k \ \big(\pi(0) = s \ \ land\ \pi(1) \models \alpha\big), \\
s \models \mathrm{EG}\alpha & \textit{iff} & \begin{cases} \exists \pi \in P_k\big(\pi(0) = s \ \ and\ \forall_{0 \le j \le k} \pi(j) \models \alpha\big), & if\ loop(\pi) \neq \emptyset, \\ false, & otherwise, \end{cases} \\
s \models \mathrm{E}(\alpha \mathrm{U} \beta) & \textit{iff} & \exists \pi \in P_k\big(\pi(0) = s \ and\ \exists_{0 \le j \le k}\big(\pi(j) \models \beta \ and\ \forall_{0 \le i < j} \pi(i) \models \alpha\big)\big), \\
s \models \overline{\mathrm{K}}_i \alpha & \textit{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \ and\ \exists_{0 \le j \le k}(l_i(s) = l_i(\pi(j)) \ and\ \pi(j) \models \alpha)\big), \\
s \models \overline{\mathrm{D}}_\Gamma \alpha & \textit{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \text{ and } \exists_{0 \le j \le k}\big(\pi(j) \models \alpha \text{ and } s \sim_\Gamma^D \pi(j)\big)\big), \\
s \models \overline{\mathrm{E}}_\Gamma \alpha & \textit{iff} & \exists \pi \in P_k\big(\pi(0) = \iota \text{ and } \exists_{0 \le j \le k}\big(\pi(j) \models \alpha \text{ and } s \sim_\Gamma^E \pi(j)\big)\big), \\
s \models \overline{\mathcal{C}}_\Gamma \alpha & \textit{iff} & s \models \bigvee_{i \le k} (\overline{\mathrm{E}}_\Gamma)^i \alpha,
\end{array}
$$

The definition above is an extension to the knowledge modalities of the definition of [PWZ02].

Definition 8 (Validity for Bounded Semantics) *An* **ECTLK** *formula φ is valid in a k-model M_k (denoted $M \models_k \varphi$) iff $M_k, \iota \models \varphi$.*

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

Theorem 1 *Let* $\mathrm{M} = ((W, T, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$ *be a model, φ be an* **ECTLK** *formula and $k = |M|$. Then, $M \models \varphi$ iff $M \models_k \varphi$.*

It is important to mention that it is often the case that if $\mathrm{M} \models \varphi$, then there exists $k < |M|$ such that $M \models_k \varphi$. This is, in fact, the reason that the BMC method is sometimes very efficient.

5 The BMC algorithm for ACTLK

In this section we present a method of BMC for **ACTLK**. This is an extension of the method of [PWZ02]. We refer to this paper for more details. The main idea is that we can check φ over M_k by checking the satisfiability of the propositional formula $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$, where the former represents (part of) the model under consideration and the latter a number of constraints that must be satisfied on M_k for φ to be satisfied. This enables us to use the most efficient algorithms for solving the SAT-problem, i.e., the problem of the satisfiability of propositional formulas. Usually, we start with $k := 1$ and increase k by one as long as $[M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$ becomes satisfiable or k reaches $|M|$.[5] In order to do this, we have to encode the transitions in the interpreted system under consideration.

We assume that each $L_i \subseteq \{0,1\}^{n_i}$, where $n_i = \lceil log_2(|L_i|) \rceil$. So, each global state can be represented by a vector of state variables $w = (w[1], \ldots, w[n])$ (called a *global state variable*), where each $w[i]$ for $i = 1, \ldots, n$ is a vector of propositional variables of length n_i.

Moreover, 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 a number of symbolic $k-$paths. Intuitively this number depends on the formula φ under investigation. We use elements of a set LL^φ to pick indices for the various paths under consideration. We refer to [PWZ02] for more details.

To construct $[M, \varphi]_k$, we first define a propositional formula $[M^{\varphi,\iota}]_k$ that constrains $|LL^\varphi|$ symbolic k-paths to be valid k-computations of M_k. For $j \in LL^\varphi$, 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. Then, we translate the **ECTLK** formula φ into a propositional formula $[\varphi]_{M_k}$ that constrains the set of $|LL^\varphi|$ symbolic k-paths to satisfy φ. Details of the translation are given as follows.

Let $\mathcal{PV}$ be a set of propositional variables, $\mathcal{FORM}$ be a set of propositional formulas over $\mathcal{PV}$, and let $lit : \{0,1\} \times \mathcal{PV} \to \mathcal{FORM}$ be a function defined as follows: $lit(0,p) = \neg p$ and $lit(1,p) = p$. Furthermore, let w, v be global state variables. We define the following propositional formulas:

- $I_s(w) := \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n_i} lit(s[i][j], w[i][j])$, encodes the state s of the model M,
- $p(w)$ iff $p \in \mathcal{V}(w)$, where $p \in \mathcal{PV}$,

[5] This bound can be frequently improved by checking whether all the states have been already searched

- $H(w,v) := \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n_i} w[i][j] \Leftrightarrow v[i][j]$,
- $H_i(w,v) := \bigwedge_{j=1}^{n_i} w[i][j] \Leftrightarrow v[i][j]$,
- $T(w,v)$ iff $(w,v) \in T$,
- $L_{k,j}(l) := T(w_{k,j}, w_{l,j})$, codes a backward loop from the k-th state to the l-th state in the symbolic $k-$computation j, for $0 \leq l \leq k$.

Definition 9 (Unfolding of the Transition Relation) *Let $M_k = ((W, P_k, \iota), \mathcal{V})$ be the $k-$model of M, $\iota \in W$, and φ be an* ECTL *formula. The propositional formula $[M^{\varphi,\iota}]_k$ is defined as follows:*

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

where $w_{0,0}$ and $w_{i,j}$ for $i = 0, \ldots, k$ and $j \in LL^\varphi$ are global state variables.
$[M^{\varphi,\iota}]_k$ constrains all the $|LL^\varphi|$ symbolic k-paths to be valid k-computations in M_k.

The next step of our algorithm is to translate an **ECTLK** formula φ into a propositional formula. We assume that ι is the initial state of the model and use $[\varphi]_k^{[m,n]}$ to denote the translation of an **ECTLK** formula φ at $w_{m,n}$ to a propositional formula.

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

We are now in the position of checking φ over M_k by checking the satisfiability of the propositional formula $[M^{\varphi,\iota}]_k \wedge [\varphi]_k^{[0,0]}$. The translation presented above can be shown to be correct and complete.

6 Analysis of the bit transmission problem

The model checking method does not work for the whole language of **CTLK**. This follows from the fact that only validity of existential formulas can be translated into the satisfiability problem. Therefore, we can prove the validity of **ECTLK** formulas or alternatively, prove that **ACTLK** formulas are not valid.
To show an example of the procedure above we discuss the bit transmission problem [FHMV95] a typical example from the MAS literature in which temporal epistemic properties need to be checked against some protocol and a transition structure.
The bit-transmission problem involves two agents, a *sender* S, and a *receiver* R, communicating over a faulty communication channel. The channel may drop messages but will not flip the value of a bit being sent. S wants to communicate some information — the value of a bit for the sake of the example — to R.
To this end, S immediately starts sending the bit to R, and continues to do so until it receives an acknowledgement from R. R does nothing until it receives the bit; from then on it sends acknowledgements of receipt to S. S stops sending the bit to R when it receives an acknowledgement. Note that R will continue sending acknowledgments even after S has received its acknowledgement. Intuitively S will know for sure that the bit has been received by R when it gets an acknowledgement from R. R, on the other hand, will never be able to know whether its acknowledgement has been received since S does not answer the acknowledgement.

What we would like to do is to check mechanically that the protocol above guarantees that when sender receives the acknowledgement (represented by the proposition **reckack**) it then knows that the receiver knows the value of the bit. The following formula expresses the above for the value of bit equal 0 represented by the proposition $\mathbf{bit} = \mathbf{0}$:

$$\mathrm{AG}(((\mathbf{bit} = \mathbf{0}) \wedge \mathbf{reckack}) \rightarrow K_S K_R(\mathbf{bit} = \mathbf{0})).$$

First, we negate the formula and get:

$$\varphi := \mathrm{EF}(((\mathbf{bit} = \mathbf{0}) \wedge \mathbf{reckack}) \wedge \overline{\mathrm{K}}_S \overline{\mathrm{K}}_R \neg(\mathbf{bit} = \mathbf{0})).$$

The sender S can be modelled by considering four possible local states. They represent the value of the bit S is attempting to transmit, and whether or not S has received an acknowledgement from R. Three different local states are sufficient to capture the state of R: the value of the received bit, and ϵ representing a circumstance under which no bit has been received yet. So we have

$$L_S = \{\mathbf{0}, \mathbf{1}, (0, ack), (1, ack)\}, \quad L_R = \{\mathbf{0}, \mathbf{1}, \epsilon\}.$$

Given this, we can use two bits to binary represent the local states of S, and other two for R. Therefore, for L_S we have the following representation: $\mathbf{0} = (0,0), \mathbf{1} = (1,0), (0, ack) = (0,1), (1, ack) = (1,1)$, whereas for L_R we take $\mathbf{0} = (0,0), \mathbf{1} = (0,1), \epsilon = (1,0)$.
So, the global states can be represented by global state variables of the form $(w[1], w[2], w[3], w[4])$. The translation of the two propositions used in φ is as follows: $\mathbf{reckack}(w) := w[2]$ and $\mathbf{bit} = \mathbf{0}(w) := \neg w[1]$. To proceed with the translation, the first thing we need to translate is the initial state $\iota = (\mathbf{0}, \epsilon)$, where ι is binary represented by $(0,0,1,0)$. With the representation above this will be encoded by the propositional formula $I_\iota(w_{0,0}) := \neg w_{0,0}[1] \wedge \neg w_{0,0}[2] \wedge w_{0,0}[3] \wedge \neg w_{0,0}[4]$. 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 $k = 1$. We know[6] that $|LL^\varphi| = 3$ for φ, so we need to exploit three symbolic paths for the translation. Assume that $LL^\varphi = \{1,2,3\}$. Here we show only the formula $T(w_{0,1}, w_{1,1})$ representing the first transition of the first path. The remaining formulas $T(w_{0,2}, w_{1,2})$ and $T(w_{0,3}, w_{1,3})$ are defined similarly.
Consider then the joint action $(sendbit(0), \lambda)$, this generates two transitions: one from $(0, \epsilon)$ to $(0, \epsilon)$, and one from $(0, \epsilon)$ to $(0, 0)$[7]. The corresponding formula is:
$(\neg w_{0,1}[1] \wedge \neg w_{0,1}[2] \wedge w_{0,1}[3] \wedge w_{0,1}[4] \wedge \neg w_{1,1}[1] \wedge \neg w_{1,1}[2] \wedge w_{1,1}[3] \wedge \neg w_{1,1}[4]) \vee (\neg w_{0,1}[1] \wedge \neg w_{0,1}[2] \wedge w_{0,1}[3] \wedge w_{0,1}[4] \wedge \neg w_{1,1}[1] \wedge \neg w_{1,1}[2] \wedge \neg w_{1,1}[3] \wedge \neg w_{1,1}[4])$.
After simplification this corresponds to the formula:
$T(w_{0,1}, w_{1,1}) := \neg w_{0,1}[1] \wedge \neg w_{0,1}[2] \wedge w_{0,1}[3] \wedge w_{0,1}[4] \wedge \neg w_{1,1}[1] \wedge \neg w_{1,1}[2] \wedge \neg w_{1,1}[4]$. To encode the whole example we should model all the transitions. We do not do it here.
Let us now encode the formula φ we would like to check.
$[\varphi]_1^{[0,0]} := \bigvee_{i \in LL^\varphi} (\bigwedge_{l=1}^{4} (w_{0,0}[l] \Leftrightarrow w_{0,i}[l]) \ \wedge\ \bigvee_{j=0}^{1} [\alpha]_1^{[j,i]})$, where $\alpha = ((\mathbf{bit} = \mathbf{0}) \wedge reckack) \wedge \overline{\mathrm{K}}_S \overline{\mathrm{K}}_R \neg(\mathbf{bit} = \mathbf{0})$.
Next, $[\alpha]_1^{[j,i]} := \neg w_{j,i}[1] \wedge w_{j,i}[2] \wedge [\beta]_1^{[j,i]}$, where $\beta = \overline{\mathrm{K}}_S \overline{\mathrm{K}}_R \neg(\mathbf{bit} = \mathbf{0})$.

$$[\beta]_1^{[j,i]} := \bigvee_{i' \in LL^\varphi} \Big(\neg w_{0,i'}[1] \wedge \neg w_{0,i'}[2] \wedge w_{0,i'}[3] \wedge \neg w_{0,i'}[4] \wedge \bigvee_{j'=0}^{1} ([\overline{\mathrm{K}}_R \neg(\mathbf{bit} = \mathbf{0})]_1^{[j',i']} \wedge \bigwedge_{l=1}^{2} (w_{j,i}[l] \Leftrightarrow w_{j',i'}[l]))\Big).$$

$$[\overline{\mathrm{K}}_R \neg(bit = 0)]_1^{[j',i']} := \bigvee_{i'' \in LL^\varphi} \Big(\neg w_{0,i''}[1] \wedge \neg w_{0,i''}[2] \wedge w_{0,i''}[3] \wedge \neg w_{0,i''}[4] \wedge \bigvee_{j''=0}^{1} (w_{j'',i''}[1] \ \wedge\ \bigwedge_{l=3}^{4} (w_{j',i'}[l] \Leftrightarrow w_{j'',i''}[l]))\Big).$$

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

7 Conclusions

We have extended the BMC technique to temporal epistemic formulas. We have sketched how the bit transmission problem can be solved by means of this method. The next step is to implement our algorithm.

[6]The method of computing the size of LL^φ will be shown in the full version of this paper.
[7]In practice one would encode separately the local transitions of each agent together with encoding the required synchronization.

References

[BCCZ99] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of TACAS'99*, volume 1579 of *LNCS*, pp. 193–207. Springer-Verlag, 1999.

[CBRZ01] 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.

[CGP99] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[EC82] 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.

[Eme90] E. A. Emerson. *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter Temporal and Modal Logic, pp. 995–1067. Elsevier, 1990.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[MP92] Z. Manna and A. Pnueli. *Temporal Logic of Reactive and Concurrent Systems*. Barnes and Noble. Springer-Verlag, 1992.

[MMZZM01] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. of the 38th Design Automation Conference (DAC01)*, pp. 530–535, June 2001.

[MS99] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect knowledge. In *Proc. of FST&TCS*, volume 1738 of *LNCS*, pp. 432–445, 1999.

[PWZ02] W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, Vol. 51(1,2), pp. 135–156, 2002.

[vdHW02a] 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*, pp. 95–111. Springer-Verlag, 2002.

[vdHW02b] 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)*, July 2002. to appear.

Pracę zgłosił: Prof. dr hab. Antoni Mazurkiewicz

Adresy autorów: Wojciech Penczek
Institute of Computer Science, PAS
Ordona 21,
01-237 Warsaw, Poland
and
Podlasie Academy
Institute of Informatics, Siedlce, Poland
email: penczek@ipipan.waw.pl

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

Symbole klasyfikacji rzeczowej: D.2.4, F.3.1, F.4.1
Na prawach rękopisu
Printed as a manuscript

Nakład 140 egzemplarzy. Papier kserograficzny klasy III. Oddano do druku w lipcu 2002. Wydawnictwo IPI PAN. ISSN 0138-06-48.