Bounded Model Checking for Interpreted Systems: Preliminary Experimental Results*

A. Lomuscio[1], T. Łasica[2], and W. Penczek[23]

[1] Department of Computer Science
King's College London, London WC2R 2LS, United Kingdom
email: alessio@dcs.kcl.ac.uk
[2] Institute of Computer Science, PAS
01-237 Warsaw, ul. Ordona 21, Poland
email: tlasica@life.pl, penczek@ipipan.waw.pl
[3] Podlasie Academy, Institute of Informatics, Siedlce, Poland

Abstract. We present experimental results relating to a bounded model checking algorithm applied to the attacking generals problem. We use interpreted systems semantics and a logical language comprising knowledge and time.

1 Introduction

The field of MAS is concerned with the study of open, distributed systems, where the entities (processes or *agents*) show highly flexible and autonomous behaviour. MAS research spans from theoretical foundations to architectures, and applications. The problems and techniques used in MAS have much in common with the literature of distributed systems and software engineering, but contrary to these disciplines, the emphasis here is on the prominence given to concepts such as knowledge, beliefs, obligations, etc., that are used to model the agents in the system. Since information technology is facing the task of delivering ever more complex distributed applications, MAS researchers argue that much is to be gained from an approach that focuses on high-level macroscopic characteristics of the entities, at least in the modelling stage. These considerations are not new by any means, and indeed represent the traditional response that computer science offers when the intrinsic complexity of the application increases.

MAS theories involve the formal representation of agents' behaviour and attitudes. To this end various modal logics have been studied and developed, including logics for knowledge, beliefs, actions, obligations, intentions, as well as

* 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). The first author also acknowledges support from the UK Royal Academy of Engineers for funding to present this paper to the FAABS workshop, organised by NASA.

combinations of these with temporal operators. These logics are seen as *specifications* of particular classes of MAS systems. Their aim is to offer a precise description of the mental or behavioural properties that a MAS should exhibit in a specific class of scenarios.

While these investigations are conceptually valuable, they can seldom be applied *in practice*. In particular, the computational complexity of such logics is so hard [HV86] that current theorem provers seem to offer no easy solution to the problem of verification of MAS. Following a very influential paper by Halpern and Vardi [HV91], attempts have been made to use model checking techniques [CGP99] to tackle the verification problem of MAS. In particular, interpreted system semantics has been exploited to verify simple epistemic properties of MAS. Specifically, [vdHW02a,vdHW02b] analyse respectively the application of SPIN and MOCHA to model checking of LTL and ATL extended by epistemic modalities, whereas [vdMS99] studies the complexity of the model checking problem for systems for knowledge and time.

One recent attempt by the authors of this paper has involved extending bounded model checking [PWZ02] to knowledge and time [PL03a]. Bounded model checking, originally proposed in [BCCZ99,CBRZ01], is a technique based on SAT translation, that attempts to ease the problem of state-explosion by exploring only a part of the model that is sufficient to validate the particular formula that needs to be checked. The aim of this paper is to evaluate the technique presented in [PL03a], by reporting and commenting upon experimental results based on a typical MAS scenario.

The rest of the paper is organised as follows. Section 2 fixes the notation of the basic formal concepts on which this paper is based upon. Section 3 describes a BMC algorithm for **ECTLK**. In Section 4 we present an implementation for the bounded model checking algorithm. There we also discuss experimental results concerning an example in the MAS literature: the "attacking generals problem". In the final section we point to future and related work.

2 Basic concepts and notation

We assume knowledge of interpreted systems semantics [FHMV95], and the bounded checking algorithm developed for it in [PL03a]. What follows serves the purpose of fixing the notation, and providing a brief summary of the two above publications.

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 $P_i : L_i \to 2^{Act_i}$, for $i = 1, \ldots, n$, representing the functioning behaviour of every agent, and a function $P_e : L_e \to 2^{Act_e}$ for the environment. Note that this defines a non-deterministic system. We can model the computation taking place in the system by means of a transition function $t : G \times Act \to G$, where

$Act \subseteq Act_1 \times \ldots \times Act_n \times Act_E$ is the set of joint actions. Intuitively, given an initial state ι, 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. *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 set of reachable global states for the system (henceforth called simply "states"),*
- *$T \subseteq W \times W$ is a total 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 $s \sim_i s'$ iff $l_i(s') = l_i(s)$, where the function $l_i : W \to L_i$ 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,*
- *$\mathcal{V} : W \longrightarrow 2^{\mathcal{PV}}$ 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.*

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 is $\overline{\mathrm{K}}_i\alpha$, for $i \in A$,*
- *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: $\mathrm{F}\alpha \stackrel{def}{=} \mathbf{true}\mathrm{U}\alpha$, $\mathrm{A}(\alpha\mathrm{R}\beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\alpha\mathrm{U}\neg\beta)$, $\mathrm{AX}\alpha \stackrel{def}{=} \neg\mathrm{EX}\neg\alpha$, $\mathrm{AG}\alpha \stackrel{def}{=} \neg\mathrm{EF}\neg\alpha$, $\mathrm{D}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathrm{D}}_\Gamma\neg\alpha$, $\mathcal{C}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathcal{C}}_\Gamma\neg\alpha$, $\mathrm{E}_\Gamma\alpha \stackrel{def}{=} \neg\overline{\mathrm{E}}_\Gamma\neg\alpha$. Moreover, $\alpha \to \beta \stackrel{def}{=} \neg\alpha \vee \beta$. We omit the subscribe Γ of the epistemic modalities if $\Gamma = A$, i.e., Γ is the set of all the agents.

The logic **ECTLK** is the restriction of **CTLK** such that negation can be applied only to elements of $\mathcal{PV}$ — the definition of **ECTLK** is identical to Definition 2 except for $\neg p$ replacing $\neg\alpha$ in the second itemised paragraph.

The logic **ACTLK** is the restriction of **CTLK** such that its language is defined as $\{\neg\varphi \mid \varphi \in \textbf{ECTLK}\}$. It is easy to see that **ACTLK** formulas can be written as follows: $\mathrm{AX}\alpha$, $\mathrm{A}(\alpha \mathrm{R} \beta)$, $\mathrm{AF}\alpha$, $\mathrm{K}_i\alpha$, $\mathrm{D}_\Gamma\alpha$, $\mathcal{C}_\Gamma\alpha$, and $\mathrm{E}_\Gamma\alpha$.

Satisfaction and validity for **CTLK** is defined as standard [FHMV95,PL03a]. These can be defined also on a bounded model. This is essentially a temporal epistemic model as above, where the computational paths have been truncated at length k; we call them the k-computations.

Definition 3 ($k-$model). *Let* $\mathrm{M} = (\mathcal{K}, \mathcal{V})$ *be a model and* $k \in \mathbb{N}_+$. *A* $k-$*model for* M *is a structure* $\mathrm{M}_k = ((W, P_k, \sim_1, \ldots, \sim_n, \iota), \mathcal{V})$*, where* P_k *is the set of all the* k*-computations of* M*, i.e.,* $P_k = \bigcup_{s \in W} \Pi_k(s)$.

Satisfaction for the temporal operators in the bounded case depends on whether or not the computation π defines a loop, i.e., there is a transition from the last state of π to its earlier one. We refer to [PL03a] for more details.

The model checking problem ($M \models \varphi$) can be reduced to the bounded model checking problem ($M \models_k \varphi$) (see [PL03b] for details.)

3 A BMC algorithm for ECTLK

In [PL03a] a method based on bounded semantics for a temporal epistemic language defined above was presented. The main idea of the method is that we can check φ over M_k by checking the satisfiability of a propositional formula $[M, \varphi]_k = [M^{\varphi,\iota}]_k \wedge [\varphi]_{\mathrm{M}_k}$, 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 **ECTLK** formula can be done by means of a SAT-checker. Although from a theoretical point of view the complexity of this operation is no easier, in practice the efficiency of modern SAT-checkers makes the process worthwhile in many instances. In this process, an important decision to take is the size k of the truncation. We do not discuss this issue in this paper, but we do point out the fact that there are heuristics that can be applied in particular classes of examples.

A trivial mechanism, for instance, would be to start with $k := 1$, test SAT-satisfaction for the translation, and increase k by one either until $[M^{\varphi,\iota}]_k \wedge [\varphi]_{\mathrm{M}_k}$ becomes satisfiable or k reaches $|M|$, i.e., the number of states of M.

Definition 4. *BMC algorithm for* **ECTLK***:*

- *Let* $\varphi = \neg\psi$ *(where* ψ *is an* **ACTLK** *formula).*
- *Iterate for* $k := 1$ *to* $|M|$.
- *Select the* $k-$*model* M_k.
- *Select the submodels* M'_k *of* M_k *with* $|P'_k| \leq f_k(\varphi)$.
- *Translate the transition relation of the* $k-$*computations of all of the submodels* M'_k *into a propositional formula* $[M^{\varphi,\iota}]_k$.
- *Translate* φ *over all* M'_k *into a propositional formula* $[\varphi]_{\mathrm{M}_k}$.

– *Check the satisfiability of* $[M,\varphi]_k := [M^{\varphi,\iota}]_k \wedge [\varphi]_{\mathrm{M}_k}$.

The framework described in the previous sections allows us to verify the temporal epistemic properties of MAS. In principle, by means of BMC on **CTLK** 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.

In practice the technique above is most useful when the following prerequisites are observed. First we should be able to specify fully the system under consideration. This can be done for instance by giving its complete description in terms of interpreted systems, i.e., by spelling out the sets of local states, actions, protocols, and transition relations. In this way we can build the model in an automatic way (details of how this can be done are not presented in this paper). Second, the benefits of the BMC machinery are more evident when the task is to check:

1. that an **ACTLK** formula is false (on an interpreted system),
2. that an **ECTLK** formula is true (on an interpreted system).

We perform 1) when we would like to check the model for faults, i.e., we would check whether some particular formula is actually false in the model. We perform 2) when we would like to check whether the model provides for a realisation of a formula.

4 Implementation and Experimental Results

In this section we introduce an implementation of the algorithm above as well as present and evaluate some experimental results obtained with it.

The tool, BMCIS (Bounded Model Checking for Interpreted Systems), is an extension of BBMC [PWZ02]. BMCIS is a bounded model checker for **ECTLK**. It computes translations into propositional formulas mentioned in the section above, and checks them for satisfaction by means of a SAT-checker. More precisely, BMCIS takes as input an interpreted system, an **ECTLK** formula to be checked, and an integer k representing the length of the model to be generated. The output is a propositional formula that is given to the Zchaff SAT-Checker [Zha01] to test for satisfaction. If the translated formula is not satisfied, k is increased until either the translated formula is satisfiable, or k reaches the size of the model.

In the description of the interpreted system, the user is required to provide the set of local states, actions, and a protocol for each agent of the system. Synchronisation among agents is achieved by locking each local transition to a global transition for the whole system. To describe state of affairs of the system, local propositions, i.e., propositions whose truth value depend on the local state of some agent, are used. Since we are concerned with modelling properties of the agents we do not see this as a limitation.

BMCIS is written in C++ making use of STL libraries. It runs both on Unix, and Windows machines. The tests presented below were produced by using a workstation equipped with a Pentium III 1GHz and 512 RAM under Linux RedHat 7.3.

4.1 Attacking generals

We now give experimental results for an often analysed scenario: the coordinated attack problem. This is an example discussed in MAS, in distributed computing, as well as in epistemic logic. It concerns coordination of agents in the presence of unreliable communication. It is also known as the coordinated attack problem:

> Two divisions of an army, each commanded by a general, are camped on the hilltops overlooking a valley. In the valley awaits the enemy. It is clear that if both divisions attack the enemy simultaneously, they will win the battle. While if one division attacks, it will be defeated. As a result neither general will attack unless he is absolutely sure the other will attack with him. In particular, one general will not attack if he receives no messages. The commander of the first division wishes to coordinate a simultaneous attack (at some point the next day). The generals can only communicate by means of messengers. Normally it takes a messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will it take them to coordinate an attack?
> ([FHMV95] page 176).

This example is appealing for at least two reasons. First, it is an instance of a recurring problem in coordination for action in MAS. Second, it can be formally analysed by means of interpreted systems and temporal epistemic logic. Crucially two key properties can be proven about the scenario above.

- No general will attack before it is common knowledge that they will both attack.
- No joint protocol can establish common knowledge, unless the delay with which the messages may be delivered is bounded.

From this one can infer that the generals will not attack on the night, even though the messengers do deliver the messages in exactly one hour. We refer to the literature [HM90] for a comprehensive analysis of the example, which turns out to be more subtle than it may appear at first. What we point out here is that the problem resides with the agents being forced to contemplate the possibility of the messenger getting lost at each round. This makes it impossible for common knowledge to be obtained in this circumstance[1].

Obviously it is problematic to perform model checking on the scenario as described above. The reason is that it is, in fact, a description for a *family* of joint protocols for the generals (the question of how long it will take to coordinate is more technically posed as "what joint protocol should the generals be running"). Indeed it looks difficult to prove in any way other than analytically as in the literature impossibility results of the kind mentioned above.

For the purpose of this paper, we chose a particular joint protocol for the scenario above and verify the truth and falsity of particular formulas that capture its key characteristics. The variant we analyse is the following:

> After having studied the opportunity of doing so, general A may issue a request-to-attack order to general B. A will then wait to receive an acknowledgement from B, and will attack immediately after having received it. General B will not issue request-to-attack orders himself, but if his assistance is requested, he will acknowledge the request, and will attack after a suitable time for his messenger to reach A (*assuming no delays*) has elapsed. A joint attack guarantees success, and any non-coordinated attack causes defeat of the army involved.

We can model the example above with the interpreted system which results from the product of the local protocols for *general A*, *general B*, and *the environment*. These are shown in Figures 1, 2, and 3, respectively. The global states are defined as 3-tuples of the local states of the above agents, whereas the global transitions are obtained by synchronising the local transitions with the same labels. The agent, which does not contribute to the synchronisation is assumed not to change its present local state.

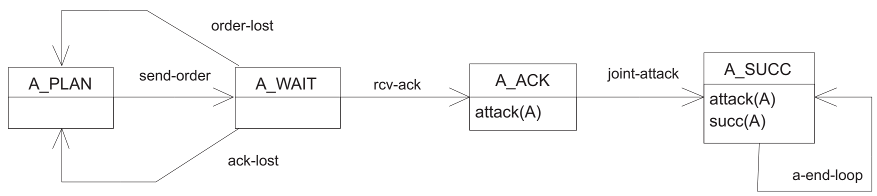

Fig. 1. Attacking generals: the local transition structure for general A

Figure 1 shows the transition structure for general A. *General A* begins by sending a request-to-attack order to *general B*, where the environment is used

[1] One should not infer from this, as it is sometimes mistakenly done, that common knowledge can *never* been achieved by message passing. The key element here is that messages may be delayed without a bound.

as a communication channel. After sending the order, A waits for the acknowledgement from B to come. Three things may happen at this stage. Either the order or the acknowledgement sent by B may be lost (this is represented by the transitions labelled *order-lost* and *ack-lost*, resp.). In this case, A goes back to state *A_PLAN*. If the message does get through together with the acknowledgement, A moves to state *A_ACK*. The proposition **attack(A)** is true in that state meaning that *general A* will attack the enemy.

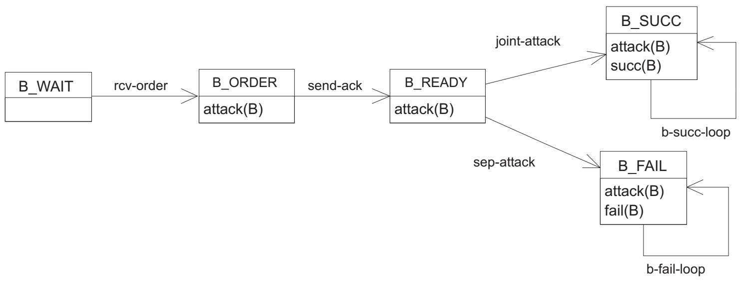

Fig. 2. Attacking generals: the local transition structure for general B

Figure 2 shows the transition structure for general B. *General B* begins by receiving a request-to-attack order from *general A* (represented by transition labelled *rcv-order*). In fact, B receives the order from the environment, which is used as a communication channel. After receiving the order, B sends an acknowledgement to A and moves to state *B_READY*. The proposition **attack(B)** is true in that state meaning that *general B* will attack the enemy. Two things may happen at this stage.

Agent B may attack on its own. This occurs when the acknowledgement is lost, and results in a transition of B to state *B_FAIL* (represented by transition labelled *sep-attack*). B's attack is joint with A's; this occurs when the acknowledgement does get through, and results in B moving to state *B_SUCC* (represented by transition labelled *joint-attack*).

Figure 3 shows the transition structure for the environment, which may deliver messages or lose them. The initial local state is *WAIT*. After receiving the order from A (represented by transition labelled *send-order*), the environment may pass it to B (transition labelled *rcv-order*) or lose it (transition labelled *lost-order*), and moves to state *WAIT*. There, after receiving an acknowledgement from B (represented by transition labelled *send-ack*), the environment goes to state *ACK*. Notice that this only happens where B has received the order, i.e., B moved to state *B_ORDER*. From *ACK* two things are possible. Either the environment chooses to pass the acknowledgement to B by executing the transition labelled *rcv-ack* or to lose it by executing the transition labelled *lost-ack*, which results in reaching state *WAIT* or *ACK_LOST*, resp. If the acknowledgment

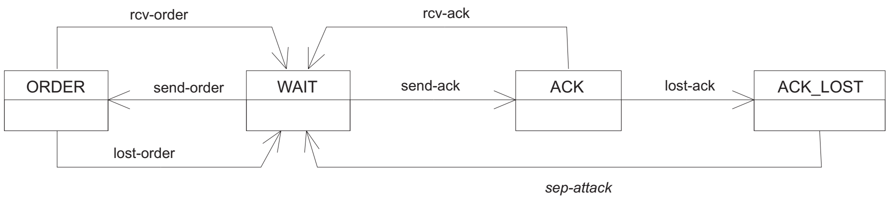

Fig. 3. Attacking generals: the local transition structure for the environment

is lost, the environment moves to state $WAIT$ (transition labelled *sep-attack*), which represents a separate attack of B.

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

1. $M \models AG(\mathbf{attack(A)} \Rightarrow K_B\mathbf{attack(A)})$,
2. $M \models AG(\mathbf{attack(B)} \Rightarrow K_A K_B\mathbf{attack(A)})$,
3. $M \models EF\mathbf{fail(B)} \wedge EF(succ(A) \wedge \mathbf{succ(B)})$,
4. $M \models EF(\mathbf{attack(B)} \wedge EG\overline{C}_{\{A,B\}}\neg\mathbf{attack(B)})$,
5. $M \models AG(\mathbf{attack(B)} \Rightarrow AF(K_B K_A\mathbf{attack(B)}))$.

In the above, the proposition $\mathbf{attack(A)}$ is true on all the global states of the model M (for the interpreted system) having A's local state equal to A_ACK or A_SUCC. The proposition $\mathbf{attack(B)}$ is true on all the states of the model M having B's local state equal to A_ACK or A_SUCC. The proposition $\mathbf{fail(B)}$ is true on all the states with B's local state equal to B_FAIL, whereas $\mathbf{succ(A)}$ on all the states where A's local state is equal to A_SUCC. Similarly, $\mathbf{succ(B)}$ holds on all the states where B's local state is equal to B_SUCC.

Property 1) states that whenever general A decides to attack, then general B knows about it. Property 2) says that if B decides to attack, then A knows that B knows that A will attack. Property 3) states that there exist (separate) evolutions leading to success and failure. Property 4) states that it is possible that A and B will never achieve the common knowledge about B's attack. Property 5) specifies that if B decides to attack, then he will always eventually know that A knows that B had decided to attack

Formulas 1,2 and 5 are not true on the interpreted system in consideration, whereas formulas 3-4 are. This means that we can show that the negations of the formulas 1,2, and 5, and formulas 3,4 (which are all **ECTLK** formulas) are valid in the model. In all of the experiments the satisfiability of the corresponding translation of an **ECTLK** formula was found to be reasonably fast.

For each tested formula the experimental results are presented in the following form:

K	the bound of the bounded model
CNF clauses	the number of the CNF clauses
BMC-memory	memory in kB used by BMCIS
BMC-time	time in seconds used by BMCIS
SAT-time	time in seconds for SAT-checking, Zchaff

The **property 1** in **ACTLK** : $AG(\mathbf{attack(A)} \Rightarrow K_B\mathbf{attack(A)})$
Its negation in **ECTLK** : $EF(\mathbf{attack(A)} \wedge \overline{K}_B\neg\mathbf{attack(A)})$

K	CNF clauses	BMC-memory	BMC-time	SAT-time	Satisfiable
1	514	68	<0.01	<0.01	N
2	909	88	0.01	<0.01	N
3	1352	116	0.01	<0.01	N
4	1843	140	0.02	<0.01	Y

The **property 2** in **ACTLK** : $AG(\mathbf{attack(B)} \Rightarrow K_AK_B\mathbf{attack(A)})$
Its negation in **ECTLK** : $EF(\mathbf{attack(B)} \wedge \overline{K}_A\overline{K}_B\neg\mathbf{attack(A)})$

K	CNF clauses	BMC-memory	BMC-time	SAT-time	Satisfiable
1	561	72	<0.01	<0.01	N
2	1023	100	0.01	<0.01	Y

The **property 3** in **ECTLK** : $EF\mathbf{fail(B)} \wedge EF(\mathbf{succ(A)} \wedge \mathbf{succ(B)})$

K	CNF clauses	BMC-memory	BMC-time	SAT-time	Satisfiable
1	909	88	0.01	<0.01	N
2	1533	120	0.02	<0.01	N
3	2157	156	0.02	<0.01	N
4	2781	188	0.03	<0.01	N
5	3405	224	0.04	<0.01	Y

The **property 4** in **ECTLK** : $EF(\mathbf{attack(B)} \wedge EG\overline{C}_{\{A,B\}}\neg\mathbf{attack(B)})$

K	CNF clauses	BMC-memory	BMC-time	SAT-time	Satisfiable
1	2113	152	0.04	<0.01	N
2	4087	256	0.29	0.02	N
3	6353	372	1.96	0.02	Y

The **property 5** in **ACTLK** : $AG(\mathbf{attack(B)} \Rightarrow AF(K_BK_A\mathbf{attack(B)}))$
Its negation in **ECTLK** : $EF(\mathbf{attack(B)} \wedge EG(\overline{K}_B\overline{K}_A\neg\mathbf{attack(B)}))$

K	CNF clauses	BMC-memory	BMC-time	SAT-time	Satisfiable
1	2173	156	0.06	<0.01	N
2	3979	248	0.22	0.02	N
3	6113	352	0.57	0.02	N
4	8575	484	1.24	0.07	N
5	11365	624	2.43	0.02	Y

5 Conclusions

While theoretical studies in MAS have long been focused on theories, i.e., *specifications* of MAS, a growing number of researchers have now started investigating the issue of concrete verification of MAS. In this work we have described an implementation of the framework of bounded model checking for knowledge and time discussed in [PL03a]. A variant of the attacking generals problem has been implemented and various formulas verified. The tables of the previous section show that the implementation performed as intended for the task. We feel the methodology of bounded model checking can tackle examples that are considerably more complex than the one explored here. Indeed, the purpose of the work presented in this paper is simply illustrative of the kind of problems we can solve. In future work we plan to test the methodology above to complex scenarios with large number of states.

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

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

[HM90] J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

[HV86] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. In *ACM Symposium on Theory of Computing (STOC '86)*, pages 304–315, Baltimore, USA, May 1986. ACM Press.

[HV91] J. Halpern and M. Vardi. *Model checking vs. theorem proving: a manifesto*, pages 151–176. Artificial Intelligence and Mathematical Theory of Computation. Academic Press, Inc, 1991.

[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*, pages 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.

[vdMS99] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect knowledge. In *Proceedings of Proc. of FST&TCS*, volume 1738 of *Lecture Notes in Computer Science*, pages 432–445, Hyderabad, India, 1999.

[PL03a] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via model checking. In T. Sandholm, editor, *Proceedings of AAMAS03*, 2003. To appear.

[PL03b] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *submitted to Fundamenta Informaticae*, 2003.

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

[Zha01] L. Zhang. Zchaff. http://www.ee.princeton.edu/~chaff/zchaff.php, 2001.