MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications

Petr Čermák[1], Alessio Lomuscio[1], Fabio Mogavero[2], and Aniello Murano[2]

[1] Imperial College London, UK
[2] Università degli Studi di Napoli Federico II, Italy

1 Introduction

Model checking has come of age. A number of techniques are increasingly used in industrial setting to verify hardware and software systems, both against models and concrete implementations. While it is generally accepted that obstacles still remain, notably handling infinite state systems efficiently, much of current work involves refining and improving existing techniques such as predicate abstraction.

At scientific level a major avenue of work remains the development of verification techniques against rich and expressive specification languages. Over the years there has been a natural progression from checking reachability only to a large number of techniques (BDDs, BMC, abstraction, etc.) catering for LTL [28], CTL [10], and CTL* [12]. More recently, ATL and ATL* [3] were introduced to analyse systems in which some components, or *agents*, can enforce temporal properties on the system. The paths so identified correspond to infinite games between a coalition and its complement. ATL is well explored theoretically and at least two toolkits now support it [4, 19, 20].

It has however been observed that ATL* suffers from a number of limitations when one tries to apply it to multi-agent system reasoning and games [1, 2, 5, 15, 17, 21, 31]. One of these is the lack of support for binding strategies explicitly to various agents or to the same agent in different contexts. To overcome this and other difficulties, *Strategy Logic* (SL) [27], as well as some useful variants of it [8, 24–26], has been put forward. Key game-theoretic properties such as *Nash equilibria*, not expressible in ATL*, can be captured in SL.

In this paper we describe the model checker MCMAS-SLK. The tool supports the verification of systems against specifications expressed in a variant of SL that includes epistemic modalities. The synthesis of agents' strategies to satisfy a given parametric specification, as well as basic counterexample generation, are also supported. MCMAS-SLK, released as open-source, implements novel labelling algorithms for SL, encoded on BDDs, and reuses existing algorithms for the verification of epistemic specifications [29].

2 Epistemic Strategy Logic

Underlying Framework. Differently from other treatments of SL, originally defined on concurrent game structures, we here define the logic on *interpreted*

systems [14]. Doing so enables us to integrate the logic with epistemic concepts. Each agent is modelled in terms of its local states (given as a set of variables), a set of actions, a protocol specifying what actions may be performed at a given local state, and a local evolution function returning a target local state given a local state and a joint action for all the agents in the system. Interpreted systems are attractive for their modularity; they naturally express systems with incomplete information, and are amenable to verification [16, 19].

Syntax. SL has been introduced as a powerful formalism to reason about various equilibria concepts in non-zero sum games and sophisticated cooperation concepts in multi-agent systems [8, 27]. These are not expressible in previously explored logics including those in the ATL* hierarchy. We here put forward an epistemic extension of SL by adding a family of knowledge operators [14].

Formulas in epistemic strategy logic, or strategy logic with knowledge (SLK), are built by the following grammar over atomic propositions $p \in \mathrm{AP}$, variables $x \in \mathrm{Vr}$, and agents $a \in \mathrm{Ag}$ ($\mathrm{A} \subseteq \mathrm{Ag}$ denotes a set of agents):

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \langle\!\langle x \rangle\!\rangle \varphi \mid (a, x)\varphi \mid \mathtt{K}_a\varphi \mid \mathtt{D}_{\mathrm{A}}\varphi \mid \mathtt{C}_{\mathrm{A}}\varphi.$$

SLK extends LTL [28] by means of an *existential strategy quantifier* $\langle\!\langle x \rangle\!\rangle$ and *agent binding* (a, x). It also includes the epistemic operators $\mathtt{K}_a$, $\mathtt{D}_{\mathrm{A}}$, and $\mathtt{C}_{\mathrm{A}}$ for individual, distributed, and common knowledge, respectively [14]. Intuitively, $\langle\!\langle x \rangle\!\rangle \varphi$ is read as *"there exists a strategy x such that φ holds"*, whereas $(a, x)\varphi$ stands for *"bind agent a to the strategy associated with the variable x in φ"*. The epistemic formula $\mathtt{K}_a\varphi$ stands for *"agent a knows that φ"*; $\mathtt{D}_{\mathrm{A}}\varphi$ encodes *"the group* A *has distributed knowledge of φ"*; while $\mathtt{C}_{\mathrm{A}}\varphi$ represents *"the group* A *has common knowledge of φ"*. Similarly to first-order languages, we use $\mathsf{free}(\varphi)$ to represent the *free agents and variables* in a formula φ. Formally, $\mathsf{free}(\varphi) \subseteq \mathrm{Ag} \cup \mathrm{Vr}$ contains *(i)* all agents having no binding after the occurrence of a temporal operator and *(ii)* all variables having a binding but no quantification. For simplicity, we here consider only formulas where the epistemic modalities are applied to sentences, *i.e.*, formulas without free agents or variables. Lifting this restriction is not problematic. To establish the truth of a formula, the set of strategies over which a variable can range needs to be determined. For this purpose we use the set $\mathsf{sharing}(\varphi, x)$ containing all agents bound to a variable x within a formula φ.

Semantics. The concepts of *path*, *play*, *strategy*, and *assignment* (for agents and variables) can be defined on interpreted systems similarly to the way they are defined on concurrent game structures. We refer to [23, 27] for a detailed presentation. Intuitively, a strategy identifies paths in the model on which a formula needs to be verified. Various variants of interpreted systems have been studied. We here adopt the memoryless version where the agents' local states do not necessarily include the local history of the run. Consequently, strategies are also memoryless. Note that this markedly differs from the previous perfect recall semantics of SL, which is defined on memoryful strategies. We consider this setting because memoryful semantics with incomplete information leads to an undecidable model checking problem [11].

Given an interpreted system $\mathcal{I}$ having G as a set of global states, a state $g \in G$, and an assignment χ defined on $\mathsf{free}(\varphi)$, we write $\mathcal{I}, \chi, g \models \varphi$ to indicate

that the SLK formula φ holds at g in $\mathcal{I}$ under χ. The semantics of SLK formulas is inductively defined by using the usual LTL interpretation for the atomic propositions, the Boolean connectives $\neg$ and $\wedge$, as well as the temporal operators $\mathtt{X}$ and $\mathtt{U}$. The epistemic modalities are interpreted as standard by relying on notions of equality on the underlying sets of local states [14]. The inductive cases for strategy quantification $\langle\langle x \rangle\rangle$ and agent binding (a, x) are given as follows. $\mathcal{I}, \chi, g \models \langle\langle x \rangle\rangle \varphi$ iff there is a memoryless strategy f for the agents in $\mathsf{sharing}(\varphi, x)$ such that $\mathcal{I}, \chi[x \mapsto \mathsf{f}], g \models \varphi$ where $\chi[x \mapsto \mathsf{f}]$ is the assignment equal to χ except for the variable x, for which it assumes the value f. $\mathcal{I}, \chi, g \models (x, a)\varphi$ iff $\mathcal{I}, \chi[a \mapsto \chi(x)], g \models \varphi$, where $\chi[a \mapsto \chi(x)]$ denotes the assignment χ in which agent a is bound to the strategy $\chi(x)$.

Model Checking and Strategy Synthesis. Given an interpreted system $\mathcal{I}$, an initial global state g_0, an SLK specification φ, and an assignment χ defined on $\mathsf{free}(\varphi)$, the *model checking problem* concerns determining whether $\mathcal{I}, \chi, g_0 \models \varphi$. Given an interpreted system $\mathcal{I}$, an initial global state g_0, and an SLK specification φ, the *strategy synthesis problem* involves finding an assignment χ such that $\mathcal{I}, \chi, g_0 \models \varphi$.

The model checking problem for systems with memoryless strategies and imperfect information against ATL and ATL* specifications is in PSpace [7]. The algorithm can be adapted to show that the same result applies to SLK. It follows that SLK specifications do not generate a harder model checking problem even though they are more expressive.

3 The Model Checker MCMAS-SLK

State Labelling Algorithm. The model checking algorithm for SLK extends the corresponding ones for temporal logic in two ways. Firstly, it takes as input not only a formula, but also a binding which assigns agents to variables. Secondly, it does not merely return sets of states, but sets of pairs $\langle g, \chi \rangle$ consisting of a state g and an assignment of variables to strategies χ. A pair $\langle g, \chi \rangle \in \mathrm{Ext}$ is called an *extended state*; intuitively, χ represents a strategy assignment under which the formula holds at state g.

Given an SLK formula φ and a binding $b \in \mathrm{Bnd} \triangleq \mathrm{Ag} \to \mathrm{Vr}$, the model checking algorithm $\mathsf{Sat}\colon \mathrm{SLK} \times \mathrm{Bnd} \to 2^{\mathrm{Ext}}$, returning a set of extended states, is defined as follows, where $a \in \mathrm{Ag}$ is an agent, $\mathrm{A} \subseteq \mathrm{Ag}$ a set of agents, and $x \in \mathrm{Vr}$ a variable:

- $\mathsf{Sat}(p, b) \triangleq \{\langle g, \chi \rangle : g \in h(p) \wedge \chi \in \mathrm{Asg}\}$, with $p \in \mathrm{AP}$;
- $\mathsf{Sat}(\neg\varphi, b) \triangleq \mathsf{neg}(\mathsf{Sat}(\varphi, b))$;
- $\mathsf{Sat}(\varphi_1 \wedge \varphi_2, b) \triangleq \mathsf{Sat}(\varphi_1, b) \cap \mathsf{Sat}(\varphi_2, b)$;
- $\mathsf{Sat}((a, x)\varphi, b) \triangleq \mathsf{Sat}(\varphi, b[a \mapsto x])$;
- $\mathsf{Sat}(\langle\langle x \rangle\rangle\varphi, b) \triangleq \{\langle g, \chi \rangle : \exists \mathsf{f} \in \mathrm{Str}_{\mathsf{sharing}(\varphi, x)}.\, \langle g, \chi[x \mapsto \mathsf{f}] \rangle \in \mathsf{Sat}(\varphi, b)\}$;
- $\mathsf{Sat}(\mathtt{X}\varphi, b) \triangleq \mathsf{pre}(\mathsf{Sat}(\varphi, b), b)$;
- $\mathsf{Sat}(\varphi_1 \mathtt{U} \varphi_2, b) \triangleq \mathrm{lfp}_X[\mathsf{Sat}(\varphi_2, b) \cup (\mathsf{Sat}(\varphi_1, b) \cap \mathsf{pre}(X, b))]$;
- $\mathsf{Sat}(\mathtt{K}_a\varphi, b) \triangleq \mathsf{neg}(\{\langle g, \chi \rangle : \exists \langle g', \chi' \rangle \in \mathsf{Sat}(\neg\varphi, \varnothing).g' \sim_a g\})$;

– $\mathsf{Sat}(\mathtt{D}_{\mathrm{A}}\varphi, b) \triangleq \mathsf{neg}(\{\langle g, \chi\rangle : \exists\, \langle g', \chi'\rangle \in \mathsf{Sat}(\neg\varphi, \varnothing).g' \sim^{\mathtt{D}}_{\mathrm{A}} g\})$;
– $\mathsf{Sat}(\mathtt{C}_{\mathrm{A}}\varphi, b) \triangleq \mathsf{neg}(\{\langle g, \chi\rangle : \exists\, \langle g', \chi'\rangle \in \mathsf{Sat}(\neg\varphi, \varnothing).g' \sim^{\mathtt{C}}_{\mathrm{A}} g\})$.

Above we use $h(p)$ to denote the set of global states where atom p is true; $\mathsf{pre}(C, b)$ is the set of extended states that temporally precede C subject to a binding b; $\mathsf{neg}(C)$ stands for the set of extended states $\langle g, \chi\rangle$ such that for each extended state $\langle g, \chi'\rangle \in C$, there is some variable $x \in \mathsf{dom}(\chi) \cap \mathsf{dom}(\chi')$, such that the strategies $\chi(x)$ and $\chi'(x)$ disagree on the action to be carried out in some global state $g' \in \mathsf{dom}(\chi(x)) \cap \mathsf{dom}(\chi'(x))$ (*i.e.*, $\chi(x)(g') \neq \chi'(x)(g')$); $\mathrm{Str}_{\mathsf{sharing}(\varphi,x)}$ is the set of strategies shared by the agents bound to the variable x in the formula φ; finally, $\sim_a$, $\sim^{\mathtt{D}}_{\mathrm{A}}$, and $\sim^{\mathtt{C}}_{\mathrm{A}}$ represent the individual, distributed, and common epistemic accessibility relations for agent a and agents A defined on the respective notions of equality of agents' local states. The set of global states of an interpreted system $\mathcal{I}$ satisfying a given formula $\varphi \in$ SLK is calculated from the algorithm above by computing $\|\varphi\|_{\mathcal{I}} \triangleq \{g \in G : \langle g, \varnothing\rangle \in \mathsf{Sat}(\varphi, \varnothing)\}$.

BDD Translation. Given an interpreted system $\mathcal{I}$ and an SLK formula φ, we now summarise the steps required to implement the labelling algorithm above using OBDDs [6]. We represent global states and joint actions as Boolean vectors $\overline{v}$ and $\overline{w}$, respectively [29]. Similarly, an assignment χ is represented as a Boolean vector $\overline{u}$ with $K = \sum_{x\in \mathrm{Vr}} \sum_{S\in G/\sim^{\mathtt{C}}_{\mathsf{sharing}(\varphi,x)}} \left\lceil \log_2 \left| \bigcap_{g\in S} \bigcap_{a\in\mathsf{sharing}(\varphi,x)} P_a(l_a(g)) \right| \right\rceil$ Boolean variables. Intuitively, for each variable $x \in$ Vr and set of shared local states $S \in G/\sim^{\mathtt{C}}_{\mathsf{sharing}(\varphi,x)}$, we store which action should be carried out. An extended state $\langle g, \chi\rangle \in$ Ext is then represented as a conjunction of the variables in $\overline{v}_g$ and $\overline{u}_\chi$.

Given a binding $b \in$ Bnd, we encode the protocol $P(\overline{v}, \overline{w})$, the evolution function $t(\overline{v}, \overline{w}, \overline{v'})$, and the strategy restrictions $S^b(\overline{v}, \overline{w}, \overline{u})$, as in [20]. The temporal transition is encoded as $R^b_t(\overline{v}, \overline{v'}, \overline{u}) = \bigvee_{\overline{w}\in Act} t(\overline{v}, \overline{w}, \overline{v'}) \wedge P(\overline{v}, \overline{w}) \wedge S^b(\overline{v}, \overline{w}, \overline{u})$. Observe that we quantify over actions, encoded as $\overline{w}$, as in [20], but we store the variable assignment in the extra parameter $\overline{u}$. Quantification over the variable assignment is performed when a strategy quantifier is encountered.

Given this, the algorithm $\mathsf{Sat}(\cdot, \cdot)$ is translated into operations on BDDs representing the encoded sets of extended states.

Implementation and Usage. The model checker MCMAS-SLK [22] contains an implementation of the procedure described previously. To do this, we took MCMAS as baseline [19]. MCMAS is an open-source model checker for the verification of multi-agent systems against ATL and epistemic operators. We used MCMAS to parse input and used some of its existing libraries for handling counter-examples, which were extended to handle SLK modalities.

MCMAS-SLK takes as input a system description given in the form of an ISPL file [19] providing the agents in the system, their possible local states, their protocols, and their evolution functions. Upon providing SLK specifications, the checker calculates the set of reachable extended states, encoded as OBDDs, and computes the results by means of the labelling algorithm described previously. If the formula is not satisfied, a counterexample is provided in the form of strategies for the universally quantified variables.

4 Experimental Results and Conclusions

Evaluation. To evaluate the proposed approach, we present the experimental results obtained on the dining cryptographers protocol [9, 19] and a variant of the cake-cutting problem [13]. The experiments were run on an Intel Core i7-2600 CPU 3.40GHz machine with 8GB RAM running Linux kernel version 3.8.0-34-generic. Table 1 reports the results obtained when verifying the dining cryptographers protocol against the specifications $\phi_{\text{CTLK}} \triangleq \texttt{AG}\psi$ and $\phi_{\text{SLK}} \triangleq \wp\texttt{G}\psi$, with $[\![x]\!]\varphi \triangleq \neg\langle\!\langle x\rangle\!\rangle\neg\varphi$, where:

$$\psi \triangleq (\text{odd} \wedge \neg \text{paid}_1) \rightarrow (\texttt{K}_{\text{c}_1}(\text{paid}_2 \vee \cdots \vee \text{paid}_n)) \wedge (\neg\texttt{K}_{\text{c}_1}\text{paid}_2 \wedge \cdots \wedge \neg\texttt{K}_{\text{c}_1}\text{paid}_n)$$

$$\wp \triangleq [\![x_1]\!] \cdots [\![x_n]\!][\![x_{\text{env}}]\!]\,(\text{c}_1, x_1) \cdots (\text{c}_n, x_n)(\text{Environment}, x_{\text{env}})$$

Table 1. Verification results for the dining cryptographers protocol.

n **crypts**	**possible states**	**reachable states**	**reachability** (s)	**CTLK** (s)	**SLK** (s)
10	3.80×10^{14}	45056	4.41	0.30	2.11
11	9.13×10^{15}	98304	1.79	0.04	5.51
12	2.19×10^{17}	212992	2.43	0.02	11.78
13	5.26×10^{18}	458752	2.17	0.11	32.41
14	1.26×10^{20}	983040	2.08	0.09	85.29
15	3.03×10^{21}	2.10×10^{6}	22.67	0.33	171.61
16	7.27×10^{22}	4.46×10^{6}	7.13	0.09	451.41
17	1.74×10^{24}	9.44×10^{6}	9.77	0.13	768.34

ϕ_{CTLK} is the usual epistemic specification for the protocol [19, 30] and ϕ_{SLK} is its natural extension where strategies are quantified. The results show that the checker can verify reasonably large state spaces. The performance depends on the number of Boolean variables required to represent the extended states. In the case of SLK specifications, the number of Boolean variables is proportional to the number of strategies (here equal to the number of agents). The last two columns of Table 1 show that the tool's performance drops considerably faster when verifying SLK formulas compared to CTLK ones. This is because CTLK requires no strategy assignments and extended states collapse to plain states. In contrast, the performance for CTLK is dominated by the computation of the reachable state space.

We now evaluate MCMAS-SLK with respect to strategy synthesis and specifications expressing Nash equilibria. Specifically, we consider a variation of the model for the classic cake-cutting problem [13] in which a set of n *agents* take turns to slice a cake of size d and the *environment* responds by trying to ensure the cake is divided fairly. We assume that at each even round the agents concurrently choose how to divide the cake; at each odd round the environment decides how to cut the cake and how to assign each of the pieces to a subset of the agents. Therefore, the problem of cutting a cake of size d between n agents is suitably divided into several simpler problems in which pieces of size $d' < d$ have to be split between $n' < n$ agents. The multi-player game terminates once each agent receives a slice.

The model uses as atomic propositions pairs $\langle i, c\rangle \in [1, n] \times [1, d]$ indicating that agent i gets a piece of cake of size c. The existence of a protocol for the cake-cutting problem is given by the following SL specification φ:

$$\varphi \triangleq \langle\!\langle x \rangle\!\rangle(\varphi_F \wedge \varphi_S), \text{where}$$

- $\varphi_F \triangleq [\![y_1]\!] \dots [\![y_n]\!](\psi_{NE} \rightarrow \psi_E)$ ensures that the protocol x is fair, *i.e.*, all Nash equilibria $(y_1, \dots, y_n)$ of the agents guarantee equity of the splitting;
- $\varphi_S \triangleq \langle\!\langle y_1 \rangle\!\rangle \dots \langle\!\langle y_n \rangle\!\rangle \psi_{NE}$ ensures that the protocol has a solution, *i.e.*, there is at least one Nash equilibrium;
- $\psi_{NE} \triangleq \bigwedge_{i=1}^{n}(\bigwedge_{v=1}^{d}(\langle\!\langle z \rangle\!\rangle \flat_i p_i(v)) \rightarrow (\bigvee_{c=v}^{d} \flat p_i(c)))$ ensures that if agent i has a strategy z allowing him to get from the environment a slice of size v once the strategies of the other agents are fixed, he is already able to obtain a slice of size $c \geq v$ by means of his original strategy y_i (this can be ensured by taking $\flat \triangleq (\text{Environment}, x)(1, y_1) \dots (n, y_n)$, $\flat_i \triangleq (\text{Environment}, x)(1, y_1) \cdots (i, z) \cdots (n, y_n)$, and $p_i(c) \triangleq \mathtt{F}\, \langle i, c\rangle$);
- $\psi_E \triangleq \flat \bigwedge_{i=1}^{n} p_i(\lfloor d/n \rfloor)$ ensures that each agent i is able to obtain a piece of size $\lfloor d/n \rfloor$ ($\flat$ and p_i are the same as in the item above).

We were able to verify the formula φ defined above on a system with $n = 2$ agents and a cake of size $d = 2$. Moreover, we automatically synthesised a strategy x for the environment (see [22] for more details). We were not able to verify larger examples; for example with $n = 2, d = 3$, there are 29 reachable states; the encoding required 105 Boolean variables (most of them represent the assignments in the sets of extended states), and the intermediate BDDs were found to be in the order of 10^9 nodes. This should not be surprising given the theoretical difficulty of the cake-cutting problem. Moreover, we are synthesising the entire protocol and not just the agents' optimal behaviour.

Conclusions. In this paper we presented MCMAS-SLK, a novel symbolic model checker for the verification of systems against specifications given in SLK. A notable feature of the approach is that it allows for the automatic verification of sophisticated game concepts such as various forms of equilibria, including Nash equilibria. Since MCMAS-SLK also supports epistemic modalities, this further enables us to express specifications concerning individual and group knowledge of cooperation properties; these are commonly employed when reasoning about multi-agent systems. Other tools supporting epistemic or plain ATL specifications exist [4,16,18,19]. In our experiments we found that the performance of MCMAS-SLK on the ATL and CTLK fragments was comparable to that of MCMAS, one of the leading checkers for multi-agent systems. This is because we adopted an approach in which the colouring with strategies is specification-dependent and is only performed after the set of reachable states is computed.

As described, a further notable feature of MCMAS-SLK is the ability to synthesise behaviours for multi-player games, thereby going beyond the classical setting of two-player games.

We found that the main impediment to better performance of the tool is the size of the BDDs required to encode sets of extended states. Future efforts will be devoted to mitigate this problem as well as to support other fragments of SL.

Acknowledgements. This research was partly funded by the EPSRC (grant EP/I00520X), the Regione Campania (Embedded System Cup project B25B09090 100007), the EU (FP7 project 600958-SHERPA), and the MIUR (ORCHESTRA project). Aniello Murano acknowledges support from the Department of Computing at Imperial College London for his research visit in July 2013.

References

1. T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-Time Temporal Logics with Irrevocable Strategies. In *Theoretical Aspects of Rationality and Knowledge'07*, pages 15–24, 2007.
2. T. Ågotnes and D. Walther. A Logic of Strategic Ability Under Bounded Memory. *Journal of Logic, Language, and Information'*, 18(1):55–77, 2009.
3. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.
4. R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Tasiran. MOCHA: Modularity in Model Checking. In *Computer Aided Verification'98*, LNCS 1427, pages 521–525. Springer, 1998.
5. T. Brihaye, A. Da Costa Lopes, F. Laroussinie, and N. Markey. ATL with Strategy Contexts and Bounded Memory. In *Logical Foundations of Computer Science'09*, LNCS 5407, pages 92–106. Springer, 2009.
6. R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *Transactions on Computers*, 35(8):677–691, 1986.
7. N. Bulling, J. Dix, and W. Jamroga. Model Checking Logics of Strategic Ability: Complexity. In *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.
8. K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. *Information and Computation*, 208(6):677–693, 2010.
9. D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1:65–75, 1988.
10. E.M. Clarke and E.A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs'81*, LNCS 131, pages 52–71. Springer, 1981.
11. C. Dima and F.L. Tiplea. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. Technical report, arXiv, 2011.
12. E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time. *Journal of the ACM*, 33(1):151–178, 1986.
13. S. Even and A. Paz. A Note on Cake Cutting. *Discrete Applied Mathematics*, 7:285–296, 1984.
14. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.
15. B. Finkbeiner and S. Schewe. Coordination Logic. In *Computer Science Logic'10*, LNCS 6247, pages 305–319. Springer, 2010.
16. P. Gammie and R. van der Meyden. MCK: Model Checking the Logic of Knowledge. In *Computer Aided Verification'04*, LNCS 3114, pages 479–483. Springer, 2004.
17. W. Jamroga and A. Murano. On Module Checking and Strategies. In *Autonomous Agents and MultiAgent Systems'14*, pages 701–708. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

18. M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Wozna, and A. Zbrzezny. VerICS 2007 - a Model Checker for Knowledge and Real-Time. *Fundamenta Informaticae*, 85(1-4):313–328, 2008.
19. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *Computer Aided Verification'09*, LNCS 5643, pages 682–688. Springer, 2009.
20. A. Lomuscio and F. Raimondi. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *Autonomous Agents and MultiAgent Systems'06*, pages 161–168. International Foundation for Autonomous Agents and Multiagent Systems, 2006.
21. A.D.C. Lopes, F. Laroussinie, and N. Markey. ATL with Strategy Contexts: Expressiveness and Model Checking. In *Foundations of Software Technology and Theoretical Computer Science'10*, LIPIcs 8, pages 120–132. Leibniz-Zentrum fuer Informatik, 2010.
22. MCMAS-SLK - A Model Checker for the Verification of Strategy Logic Specifications. http://vas.doc.ic.ac.uk/software/tools/.
23. F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. Technical report, arXiv, 2011.
24. F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic. In *Concurrency Theory'12*, LNCS 7454, pages 193–208. Springer, 2012.
25. F. Mogavero, A. Murano, and L. Sauro. On the Boundary of Behavioral Strategies. In *Logic in Computer Science'13*, pages 263–272. IEEE Computer Society, 2013.
26. F. Mogavero, A. Murano, and L. Sauro. Strategy Games: A Renewed Framework. In *Autonomous Agents and MultiAgent Systems'14*, pages 869–876. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
27. F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning About Strategies. In *Foundations of Software Technology and Theoretical Computer Science'10*, LIPIcs 8, pages 133–144. Leibniz-Zentrum fuer Informatik, 2010.
28. A. Pnueli. The Temporal Logic of Programs. In *Foundation of Computer Science'77*, pages 46–57. IEEE Computer Society, 1977.
29. F. Raimondi and A. Lomuscio. Automatic Verification of Multi-Agent Systems by Model Checking via Ordered Binary Decision Diagrams. *Journal of Applied Logic*, 5(2):235–251, 2007.
30. R. van der Meyden and K. Su. Symbolic Model Checking the Knowledge of the Dining Cryptographers. In *Computer Security Foundations Workshop'04*, pages 280–291. IEEE Computer Society, 2004.
31. D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-Time Temporal Logic with Explicit Strategies. In *Theoretical Aspects of Rationality and Knowledge'07*, pages 269–278, 2007.