Towards a Model Checking Tool for Strategy Logic with Simple Goals (short paper)*

Vadim Malvone[1] and Silvia Stranieri[2]

[1] Télécom Paris, France
vadim.malvone@telecom-paris.fr
[2] Università degli studi di Napoli Federico II, Italy
silvia.stranieri@unina.it

Abstract. In this work, we raise the need for an implementation of a model checker for Strategy Logic with Simple Goals ($\mathsf{SL}[\mathsf{SG}]$), a recently-introduced fragment of Strategy Logic (SL). Notably, $\mathsf{SL}[\mathsf{SG}]$ subsumes the logic ATL and is strictly contained in $\mathsf{SL}[\mathsf{1G}]$, a well-known fragment of SL. Thus, the model checker for $\mathsf{SL}[\mathsf{1G}]$ in MCMAS can handle $\mathsf{SL}[\mathsf{SG}]$ formulas as well. However we show that, for $\mathsf{SL}[\mathsf{SG}]$ formulas that are in ATL, one can save space and time by using the MCMAS model checker for ATL. As the model checking complexity for both $\mathsf{SL}[\mathsf{SG}]$ and ATL is PTIME-complete, there is hope that an implementation in MCMAS for $\mathsf{SL}[\mathsf{SG}]$ would work as fast as that for ATL.

Keywords: Strategy Logic · Multi-Agent Systems · Model Checking Tools

1 Introduction

In formal methods for multi-agent systems, logics for the strategic reasoning have had a major role. Among the others, ATL^* (and ATL) has come to the fore and largely explored for practical use [1]. More recently, Strategy Logic (SL)[5] has come out, where strategies are treated explicitly as first order objects and associated to agents by means of a binding operator. We recall that SL makes use of the binding operator and strategy quantifiers along its syntax. For the latter, we have the existential operator $\exists x$ and the dual universal operator $\forall x$ that can be read as "for some strategy x, ..." and "for all strategies x, ...", respectively. The binding operator (x, a) means that "by using strategy x, agent a can achieve...". SL is much more expressive than ATL^*, and able to express important solution concepts among which the Nash Equilibrium. The high expressiveness of SL comes at a price: the model-checking problem is non-elementary. This has led at looking for meaningful elementary fragments of SL, among the others $\mathsf{SL}[\mathsf{1G}]$ [6] and $\mathsf{SL}[\mathsf{SG}]$ [2].
$\mathsf{SL}[\mathsf{1G}]$ refers to SL formulas of the form $\wp\flat\phi$ where $\wp$ is a quantification prefix on strategies, $\flat$ a binding, and ϕ an LTL formula. $\mathsf{SL}[\mathsf{1G}]$

subsumes ATL^* and shares with it important features, among which a 2EXPTIME solution for the model checking problem [6] (implemented in MCMAS [3]). $\mathsf{SL[SG]}$ further restrict $\mathsf{SL[1G]}$ formulas in a way similarly as ATL restricts ATL^*. Thus, $\mathsf{SL[SG]}$ can be seen as the extension of ATL to arbitrary quantification on the agents' strategies [2]. Interestingly, $\mathsf{SL[SG]}$ can express meaningful concepts such as the Stakelberg equilibrium and coercion in voting, while providing a polynomial-time solution for the model checking problem. Notably, no direct implementation for $\mathsf{SL[SG]}$ in MCMAS has been exploited yet (since $\mathsf{SL[1G]}$ subsumes $\mathsf{SL[SG]}$, clearly MCMAS can handle $\mathsf{SL[SG]}$ formulas). In this work, we give some evidence that such an implementation should be played out, instead. Indeed, by restricting to $\mathsf{SL[SG]}$ formulas that can be translated to ATL, we show that over such formulas the standard MCMAS implementation for ATL works much faster than the one implemented for $\mathsf{SL[1G]}$.
Outline. The rest of the paper is organized as follows: in Section 2, we recall the basic concepts and results about Strategy Logic with Simple Goals, as well as the mechanisms behind the model checking process of MCMAS. In Section 3, we report the experiments made to solve $\mathsf{SL[SG]}$ and the equivalent ATL formulas over $\mathsf{SL[1G]}$ and ATL MCMAS, respectively. Finally, in Section 4, we provide the conclusions and some ideas for future developments.

2 Background

In this section, we report some basic notions about $\mathsf{SL[SG]}$ and the main features of the most largely employed tool for model checking, MCMAS.

2.1 Strategy Logic with Simple Goals

We briefly recall the syntax, and the main results for $\mathsf{SL[SG]}$ ([2]). Let AP, Ag, and Var be the sets of atomic propositions, agents, and variables, respectively. Given $A \subseteq Ag$ and $V \subseteq Var$, we define a *binding prefix* as a finite sequence $\flat \in \{(x,a) \mid a \in A \text{ and } x \in V\}^{|A|}$ such that $|\flat| = |A|$, and every agent $a \in A$ occurring exactly once in $\flat$. Contrarily, the same variable $x \in V$ can occur several times in $\flat$, allowing agents in A to use the same strategy more than once. A *quantification prefix* over the set V of variables is a finite sequence $\wp \in \{\exists x, \forall x \mid x \in V\}^{|V|}$ such that $|\wp| = |V|$ and every variable $x \in V$ occurs exactly once in $\wp$. $Qnt(V) \subset \{\exists x, \forall x \mid x \in V\}^{|V|}$ and $Bnd(A) \subset \{(x,a) \mid a \in A \text{ and } x \in Var\}^{|A|}$ denote the sets of all quantification and binding prefixes over variables in V and agents in A, respectively.

Definition 1 (Syntax of $\mathsf{SL[SG]}$). *Assuming the notion of free variable as reported in [6], given a formula ϕ in $\mathsf{SL[SG]}$, $\flat \in Bnd(Ag)$, $\wp \in Qnt(free(\flat\phi))$, and $p \in AP$, ϕ can be expressed as follows:*

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \wp\flat\, \mathsf{X}\, \phi \mid \wp\flat(\phi\, \mathsf{U}\, \phi)$$

Where X and U are the temporal next and until operators, respectively.

We conclude this section by recalling two main results for SL[SG].

Theorem 1 (Expressiveness of SL[SG] [2]). *Strategy Logic with Simple Goals has strictly greater expressive and distinguishing power than* ATL.

Theorem 2 (Complexity of SL[SG] [2]). *The model checking for Strategy Logic with Simple Goals is* PTIME-*complete.*

2.2 MCMAS

We recall that MCMAS is a tool to model check formulas over multi-agent systems. The setting is typically modeled through interpreted system, using a dedicated language to formalize it, ISPL (Interpreted System Programming Language), and it involves two types of agents: the *standard* agent and the environment one. The latter is used to describe boundary conditions and variables shared among the *standard* agents. The model checking procedure is based on binary decision diagrams (BDD, for short), commonly used to represent boolean functions in a compact manner. They consist of finite directed acyclic graphs with a unique initial node, in which each internal node is a boolean variable, and each terminal node is a truth value. The final aim is to determine, given an interpreted system $\mathcal{I}$, an initial state g, and a formula ϕ, if: $\mathcal{I}, g \models \phi$.

3 Comparison of the existing tools

The tool which is commonly used to model check multi-agent systems is MCMAS, which takes as input a MAS specification and a set of formulas to be verified. Initially, MCMAS was designed to solve formulas expressed in CTL and ATL. Later, with the expansion of Strategy Logic, and in particular with SL[1G], a new release of the model checker has been implemented, which supports SL[1G] formulas.
We tested the SL[1G] version of MCMAS against the standard ATL version of MCMAS, over SL[SG] formulas which can be translated in ATL. Notice that formulas written in SL[SG] are clearly supported by the SL[1G] version of MCMAS, since SL[SG] $\subseteq$ SL[1G]. The idea was to understand if such version of MCMAS behaves well even on the SL[SG] fragment, or if it would be the case of defining a new version for it.
Our study consists in testing the two versions of the model checker from two points of view:

1. By scaling on the number of variables;
2. By scaling on the number of agents.

In Table 1, we show the behavior of MCMAS in the standard version for ATL against the SL[1G] one, by varying the number of available variables in the *shell game*, in which we recall that the player has to guess which shell hides an object. In our setting, the players are the environment, that places the object underneath the selected shell, and the guesser who has to guess one among the available ones. We start from 200 possible

	ATL-**MCMAS**		SL[1G]-**MCMAS**	
shells	**time**	**space**	**time**	**space**
200	73,37	64M	2,03	30M
400	248,87	194M	8,327	45M
600	1263,63	410M	214,47	71M
800	1697,09	708M	230,26	109M
1000	613,094	169M	5630,94	134M
1200	8031,69	1660M	1473,87	239M
1400	9992,47	1947M	4732,29	283M

Table 1: Results of multiple executions on the shells example by varying the number of possible shells: comparison between the standard ATL MCMAS and the SL[1G] extension.

shells: the behavior of SL[1G] of MCMAS is better than the one of the standard ATL version, both in terms of time and space, by assigning a truth value to the same set of formulas in a one-magnitude-smaller time, and half of the space. Moving to 400 shells, the time needed for SL[1G] MCMAS is two magnitude smaller, and the space is 4 times less. If we give 600 shells, the difference is still high, again with a time of one magnitude smaller, and the needed space 5 times less. The trend is still the same with 800 available shells. Instead, when we reach 1000 shells, MCMAS for ATL seems to find an efficient BDD technique to solve the model checking, but the results are not confirmed by any other test with different input. Indeed, with 1200 and 1400 shells, MCMAS for SL[1G] keeps being better: even if the time results are of the same magnitude as the ATL MCMAS corresponding ones, the space needed is approximately 6 times less in both cases. In Table 2, instead, we show how the model checkers' response changes by varying the number of agents in the *voting game* [4]. We recall that, in the most simple scenario, this game involves two players: a voter and a coercer. After voting, the voter can decide to hand in proof of his vote, or not to do it. In the same way, the coercer can decide to punish the voter, or not to do it. In our setting, the coercer is modeled through the environment, while we tested the tool by increasing the number of voters. Precisely, by moving from 4 to 12 voters, the standard ATL-MCMAS behavior does not change significantly, by providing the truth value of the input formula in times and spaces that are of the same magnitude. On the contrary, with the SL[1G] extension of MCMAS both the time and the space needed for the execution are increased of one magnitude when the number of voters increases from 4 to 6. Instead, if we simply move it to 8 voters, the model checker forces its irregular termination, without assigning any truth value to the input formulas.

We provide some examples of formulas used to obtain the results just shown. For the *shell game* with 200 shells, we run the SL[1G] version of MCMAS over the following SL[SG] formula:

$$\varphi_s = \forall e \, \exists g \, (e, Environment)(g, Guesser) F \; win$$

	ATL-MCMAS		SL[1G]-MCMAS	
voters	**time**	**space**	**time**	**space**
4	0,035	9M	0,29	20M
6	0,038	9M	18,42	697M
8	0,027	9M	**aborted**	
10	0,09	10M		
12	0,08	10M		

Table 2: Results of multiple executions on the voting example by varying the number of agents involved: comparison between the standard ATL MCMAS and the SL[1G] extension.

requiring that, no matter what the strategy of the environment is, there always exists a strategy for the guesser to finally win. Then, we run the standard ATL version of MCMAS on the corresponding ATL formula:

$$\psi_s = \langle\langle g \rangle\rangle F\ win$$

where g is a coalition made by the guesser alone.
For the *voting scenario* with 4 voters, instead, the SL[SG] formula we tested is the following:

$$\varphi_v = \wp \flat\ F((vote_1^1 \rightarrow\ punish_1)\ and\ (vote_1^2 \rightarrow\ punish_2)\ and \\ (vote_1^3 \rightarrow\ punish_3)\ and\ (vote_1^4 \rightarrow\ punish_4))$$

where $\wp = \forall v_1 \forall v_2 \forall v_3 \forall v_4 \exists e$ and $\flat = (v_1, Voter_1)(v_2, Voter_2)(v_3, Voter_3)$ $(v_4, Voter_4)(e, Environment)$.
The above specifies that, no matter what the strategies of the voters are, there always exists a strategy for the environment such that if a voter votes the candidate 1 finally he will be punished. The corresponding ATL formula is:

$$[\![g_v]\!] F(\langle\langle g_e \rangle\rangle F(((vote_1^1 \rightarrow\ punish_1)\ and\ (vote_1^2 \rightarrow\ punish_2) \\ and\ (vote_1^3 \rightarrow\ punish_3)\ and\ (vote_1^4 \rightarrow\ punish_4))$$

where g_v is the coalition of voters and g_e is the environment alone.
To develop a tool for SL[SG] model checking, we expect to modify the existing ATL MCMAS in a way that makes it able to solve SL[SG] formulas. It will be needed to modify opportunely the parser to accept SL[SG] syntax, but we will also have to integrate the new model checking algorithm in the existent tool, to reflect the SL[SG] semantics.

4 Conclusions

With this work, we show that the current SL[1G] version of MCMAS is not the best solution to model check SL[SG] formulas. Indeed, our study

compares the behaviors of SL[1G] MCMAS and ATL MCMAS to solve SL[SG] formulas and the corresponding ATL formulas, respectively. The results show that as soon as the number of agents involved in the model grows, SL[1G] MCMAS does not behave well, until it stops working. On the contrary, the ATL version allows assigning a truth value to the corresponding formula in a reasonable time, which suggests an ad hoc implementation to model check SL[SG].
To enforce our result, it might be useful to develop a mechanism to translate formulas from SL[SG] to ATL in an automatic manner, also highlighting when such a translation is not applicable.

References

1. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. JACM **49**(5), 672–713 (2002)
2. Belardinelli, F., Jamroga, W., Kurpiewski, D., Malvone, V., Murano, A.: Strategy logic with simple goals: Tractable reasoning about strategies. In: 28th International Joint Conference on Artificial Intelligence (IJCAI 2019). pp. 88–94 (2019)
3. Čermák, P., Lomuscio, A., Murano, A.: Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29 (2015)
4. Jamroga, W., Knapik, M., Kurpiewski, D., Mikulski, Ł.: Approximate verification of strategic abilities under imperfect information. Artificial Intelligence **277**, 103172 (2019)
5. Mogavero, F., Murano, A., Vardi, M.: Reasoning about strategies. In: FSTTCS 10. pp. 133–144. LIPIcs 8, Leibniz (2010)
6. Mogavero, F., Murano, A., Perelli, G., Vardi, M.Y.: Reasoning about strategies: On the model-checking problem. ACM Transactions on Computational Logic (TOCL) **15**(4), 1–47 (2014)