STV+AGR: Towards Practical Verification of Strategic Ability Using Assume-Guarantee Reasoning

Damian Kurpiewski$^{1,2}$, Łukasz Mikulski$^{2,1}$, and Wojciech Jamroga$^{1,3}$

$^1$Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
$^2$Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland
$^3$Interdisciplinary Centre for Security, Reliability and Trust, SnT, University of Luxembourg, Luxembourg

Abstract

We present a substantially expanded version of our tool STV for strategy synthesis and verification of strategic abilities. The new version provides a web interface and support for assume-guarantee verification of multi-agent systems.

1 Introduction

Model checking of multi-agent systems (MAS) allows for formal (and, ideally, automated) verification of their relevant properties. Algorithms and tools for model checking of strategic abilities [1, 28, 9, 25] have been in development for over 20 years [2, 10, 6, 13, 7, 21, 8, 4, 3, 15, 20]. Unfortunately, the problem is hard, especially in the realistic case of agents with imperfect information [28, 5, 12].

In this paper, we propose a new extension of our experimental tool STV [19, 20] that facilitates compositional model checking of strategic properties in asynchronous MAS through assume-guarantee reasoning (AGR) [26, 11]. The extension is based on the preliminary results in [24], itself an adaptation of the AGR framework for liveness specifications from [22, 23].

2 Application Domain

Many important properties of MAS refer to strategic abilities of agents and teams. For example, the ATL$^*$ formula $\langle\langle\text{taxi}\rangle\rangle G \neg \text{fatality}$ says that the autonomous cab can drive in such a way that no one gets ever killed, and $\langle\langle\text{taxi, passg}\rangle\rangle F \text{destination}$
expresses that the cab and the passenger have a joint strategy to arrive at the destination, no matter what the other agents do. Another intuitive set of strategic requirements is provided by properties of secure voting systems \[27, 29\]. As shown by case studies \[16, 14, 18\] practical verification of such properties is still infeasible due to state-space and strategy-space explosion. \textbf{STV+AGR} addresses the specification and verification of such properties, as well as a user-friendly creation of models to be verified.

\section{Simple Voting Scenario}

To present the capabilities of \textbf{STV+AGR}, we designed an asynchronous version of the Simple Voting scenario \[15\]. The model consists of two types of agents, presented in Figure 1 and described below.

\textbf{Voter}. Every voter agent has three local variables:

\begin{itemize}
  \item \textit{vote}: the vote being cast (?\,1 or 2);
  \item \textit{reported}: the vote value presented to the coercer (?\,1, 2 or !), where ! means that the voter decided not to share her vote with the coercer;
  \item \textit{pstatus}: the punishment status (?\,T or F).
\end{itemize}
Each voter $i$ can also see the value of the $\text{pun}_i$ variable of the coercer.

The voter first casts her vote, then decides whether to share its value with the coercer. Finally, she waits for the coercer’s decision to punish her or to refrain from punishment.

**Coercer.** The coercer[$k$] has one local variable for each of $k$ voters:

- $\text{pun}_i$: whether the voter $i$ was punished or not ($\emptyset$, $T$ or $F$).

Moreover, he can observe the value of $\text{reported}_i$ for each voter $i$.

The coercer has two available actions per voter: to punish the voter or to refrain from punishment.

## 4 Formal Background

**Modules.** The main part of the input is given by a set of asynchronous modules inspired by [23], where local states are labelled with valuations of state variables. The transitions are valuations of input variables controlled by the other modules. The multi-agent system is defined by a composition of its modules.

**Strategies.** A strategy is a conditional plan that specifies what the agent(s) are going to do in every possible situation. Here, we consider the case of *imperfect information memoryless strategies*, represented by functions from the agent’s local states (or, equivalently, its epistemic indistinguishability classes) to its available actions. The outcome of a strategy from state $q$ consists of all the infinite paths starting from $q$ and consistent with the strategy.

**Logic.** Given a model $M$ and a state $q$ in the model, the $\mathbf{1\text{ATL}}^\ast$ formula $\langle \langle a \rangle \rangle \varphi$ holds in the pointed model $(M, q)$ iff there exists a strategy for agent $a$ that makes $\varphi$ true on all the outcome paths starting from any state indistinguishable from $q$. The semantics of coalitional abilities is analogous, for joint strategies of coalitions.

**Assume-guarantee reasoning.** The main idea is to cope with the state-space explosion by decomposing the goal $\varphi$ of coalition $C$ into local goals $\varphi_i$, $i \in C$, and verify them one by one against abstractions of each agent’s environment. An abstraction for $i$ is obtained by defining a single module, called the *assumption*, which guarantees that all the paths present in the original system have their counterparts in the composition of module $i$ and its associated assumption. Moreover, we use a distance between modules, based on shared synchronization actions, so that only “close” agents are taken into account when preparing the assumption for $i$. This way, one can deduce the existence of a joint strategy to obtain $\varphi$ from the existence of individual strategies that achieve local goals $\varphi_i$.

**Automated generation of assumptions.** The main difficulty in using assume-guarantee reasoning is how to define the right assumptions for the relevant modules. To this end, we propose an automated procedure that generates the assumptions, based on the subset of modules that are “close” the given module $M_i$. The abstraction is obtained by composing all the “close” modules, abstracting away their state labels and variables except for the ones that are input to
Table 1: Results of assume-guarantee verification the asynchronous variant of Simple Voting (times given in seconds)

\[ M_i, \text{as well as removing all their input variables which are not state variables of } M_i. \]

5 Technology

STV+AGR does explicit-state model checking. That is, the global states and transitions of the model are represented explicitly in the memory of the verification process. The tool includes the following new functionalities.

User-defined input. The user can load and parse the input specification from a text file that defines the groups of modules. The modules are local automata representing the agents. The groups define the partition for the assume-guarantee verification. Each group that describes the part of the coalition must also define the formula to be verified.

Web-based graphical interface. The generated models and the verification results are visualised in the intuitive web-based graphical interface. The GUI is implemented in Typescript and uses the Angular framework.

Evaluation. The assumption-guarantee scheme has been evaluated on the asynchronous variant of Simple Voting, using formula \( \varphi \equiv \langle \langle \text{Voter}_1 \rangle \rangle G(\neg \text{pstatus}_1 \vee \text{voted}_1 = 1) \). Note that the coalition consisted of only one agent, which made the decomposition of the formula trivial. The results are presented in Table 1. The first column describes the configuration of the benchmark, i.e., the number of voters. Then, we report the performance of model checking algorithms that operate on the explicit model of the whole system vs. assume-guarantee verification. DFS is a straightforward implementation of depth-first strategy synthesis. Apprx refers to the method of fixpoint-approximation [15]; besides the time, we also report if the approximation was conclusive.

6 Usage

The tool is available at stv.cs-htiew.com. The video demonstration of the tool is available at youtu.be/IDrmSRH1BA. Example specifications can be found at stv-docs.cs-htiew.com. The current version of STV+AGR allows to:
• Generate and display the composition of a set of modules into the model of a multi-agent system;

• Generate and display the automatic assumption, given a module and a distance bound;

• Provide local specifications for modules, and compute the global specification as their conjunction;

• Verify a $\mathbf{1ATL^*_{\neg X}}$ formula for a given system (using the verification methods available in the $\text{STV}$ package);

• Verify a $\mathbf{1ATL^*_{\neg X}}$ formula for a composition of a module and its automatic assumption (using the methods in $\text{STV}$);

• Verify a $\mathbf{1ATL^*_{\neg X}}$ formula for a composition of a module and a user-defined assumption (using the methods in $\text{STV}$);

• Display the verification result, including the relevant truth values and the winning strategy (if one exists).

7 Conclusions

Much complexity of model checking for strategic abilities is due to the size of the model. $\text{STV+AGR}$ addresses the challenge by implementing a compositional model checking scheme, called assume-guarantee verification. No less importantly, our tool supports user-friendly modelling of MAS, and automated generation of abstractions that are used as assumptions in the scheme.

Acknowledgement

The authors thank Witold Pazderski and Yan Kim for assistance with the web interface. The work was supported by NCBR Poland and FNR Luxembourg under the PolLux/FNR-CORE project STV (POLLUX-VII/1/2019), as well as the CHIST-ERA grant CHIST-ERA-19-XAI-010 by NCN Poland (2020/02/Y/ST6/00064).

References

[1] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. J. of the ACM, 49:672–713, 2002.

[2] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In Proc. of CAV’98, volume 1427 of LNCS, pages 521–525. Springer, 1998.
[3] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In Proc. of IJCAI’17, pages 91–97, 2017.

[4] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with imperfect information and public actions. In Proc. of AAMAS’17, pages 1268–1276, 2017.

[5] 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.

[6] S. Busard, C. Pecheur, H. Qu, and F. Raimondi. Improving the model checking of strategies under partial observability and fairness constraints. In Formal Methods and Software Engineering, volume 8829 of LNCS, pages 27–42. Springer, 2014.

[7] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In Proc. of CAV’14, volume 8559 of LNCS, pages 525–532. Springer, 2014.

[8] P. Cermák, A. Lomuscio, and A. Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In Proc. of AAAI’15, pages 2038–2044, 2015.

[9] K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. Inf. and Comp., 208(6):677–693, 2010.

[10] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISMGames: A model checker for stochastic multi-player games. In Proc. of TACAS’13, volume 7795 of LNCS, pages 185–191. Springer, 2013.

[11] E.M. Clarke, D.E. Long, and K.L. McMillan. Compositional model checking. In Proc. of LICS’89, pages 353–362. IEEE Comput. Soc. Press, 1989.

[12] C. Dima and F.L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. CoRR, abs/1102.4225, 2011.

[13] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In Proc. of AAAI’14, pages 1426–1432, 2014.

[14] W. Jamroga, Y. Kim, D. Kurpiewski, and P.Y.A. Ryan. Towards model checking of voting protocols in uppaal. In Proc. of E-Vote-ID’20, volume 12455 of LNCS, pages 129–146. Springer, 2020.

[15] W. Jamroga, . Knapik, D. Kurpiewski, and Ł. Mikulski. Approximate verification of strategic abilities under imperfect information. Artif. Int., 277, 2019.
[16] W. Jamroga, M. Knapik, and D. Kurpiewski. Model checking the SELENE e-voting protocol in multi-agent logics. In *Proc. of E-Vote-ID’18*, volume 11143 of *LNCS*, pages 100–116. Springer, 2018.

[17] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembinski, and A.W. Mazurkiewicz. Towards partial order reductions for strategic ability. *J. Artif. Intell. Res.*, 68:817–850, 2020.

[18] Wojciech Jamroga, Damian Kurpiewski, and Vadim Malvone. Natural strategic abilities in voting protocols. In *Proc. of STAST’20*, 2021. To appear.

[19] D. Kurpiewski, W. Jamroga, and M. Knapik. STV: Model checking for strategies under imperfect information. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 2372–2374. IFAAMAS, 2019.

[20] D. Kurpiewski, W. Pazderski, W. Jamroga, and Y. Kim. STV+Reductions: Towards practical verification of strategic ability using model reductions. In *Proc. of AAMAS’21*, pages 1770–1772. ACM, 2021.

[21] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *Int. J. Soft. Tools Tech. Trans.*, 19(1):9–30, 2017.

[22] A. Lomuscio, B. Strulo, N.G. Walker, and P. Wu. Assume-guarantee reasoning with local specifications. In *Proc. of ICFEM’10*, volume 6447 of *LNCS*, pages 204–219. Springer, 2010.

[23] A. Lomuscio, B. Strulo, N.G. Walker, and P. Wu. Assume-guarantee reasoning with local specifications. *Int. J. Found. Comput. Sci.*, 24(4):419–444, 2013.

[24] Ł. Mikulski, W. Jamroga, and D. Kurpiewski. Towards assume-guarantee verification of strategic ability. In *Proc. of AAMAS’22*, to appear.

[25] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comp. Log.*, 15(4):1–42, 2014.

[26] A. Pnueli. In transition from global to modular temporal reasoning about programs. In *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, pages 123–144. Springer, 1984.

[27] P.Y.A. Ryan. The computer ate my vote. In *Formal Methods: State of the Art and New Directions*, pages 147–184. Springer, 2010.

[28] P.Y. Schobbens. Alternating-time logic with imperfect recall. *Electr. Not. Theor. Comput. Sci.*, 85(2):82–93, 2004.
[29] M. Tabatabaei, W. Jamroga, and Peter Y. A. Ryan. Expressing receipt-freeness and coercion-resistance in logics of strategic ability: Preliminary attempt. In Proc. of PrAISe@ECAI’16, pages 1:1–1:8. ACM, 2016.