STV: Model Checking for Strategies under Imperfect Information

Damian Kurpiewski, Wojciech Jamroga, Michał Knapik
Institute of Computer Science, Polish Academy of Sciences
{damian.kurpiewski,w.jamroga,michal.knapik}@ipipan.waw.pl

ABSTRACT

We present an experimental tool for verification of strategic abilities under imperfect information, as well as strategy synthesis. The problem is well known to be hard, both theoretically and in practice. The tool, called **StraTegic Verifier (STV)**, implements several recently developed algorithms to overcome the complexity.

KEYWORDS

formal methods, alternating-time temporal logic, imperfect information, strategy synthesis, model checking

1 INTRODUCTION

As the systems around us become more complex, and at the same time more autonomous, the need for unambiguous specification and automated verification rapidly increases. *Logics of strategic reasoning* provide powerful tools to reason about various aspects of MAS [1, 3, 26, 31]. A typical property that can be expressed says that *the group of agents A has a collective strategy to enforce temporal property φ, no matter what the other agents in the system do*. Specifications in agent logics can be then used as input to *model checking*, which makes it possible to verify the correct behavior of a multi-agent system by an automated tool [9, 10, 13, 23].

Verification of strategic abilities is difficult for a number of reasons. The prohibitive complexity of model checking and strategy synthesis is a well known factor [5, 12, 25], which can be alleviated only to some degree by using symbolic data structures [4, 7, 13, 28]. Things become even harder for agents with imperfect information. The complexity ranges from **NP**–complete to undecidable [14, 31]. Even more importantly, fixpoint equivalences do not hold [6, 11], which makes the application of standard fixpoint algorithms invalid and the use of symbolic methods questionable. Most known approaches boil down to iteration over all the possible strategies [8, 24, 27]. Unfortunately, the number of available strategies is enormous.

Our team at PAS has recently developed two novel techniques that try to overcome the complexity [15, 17, 21]. In this short paper, we present an experimental tool **STV** that implements the techniques, together with a number of verification scenarios. The implementation is still preliminary (e.g., it does not provide a flexible input specification language). Still, it already allows to "play" with the verification problem, test the scalability of the new techniques, and visualize the complexity of models and strategies on intuitive benchmark scenarios.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

2 APPLICATION DOMAIN

STV is aimed at verification of strategic abilities in multi-agent systems, and synthesis of strategies that guarantee a given temporal goal. Many relevant properties of MAS refer to abilities of agents and their groups. In particular, most functionality requirements can be specified as the ability of the authorized users to achieve their goals. At the same time, many security properties can be phrased in terms of the inability of unauthorized users to compromise the system. Concrete examples include:

- Formalizations of *individual* and *group responsibility* [33, 34],
- Functionality properties for teams of *logistic robots*, operating in an industrial environment [22, 30],
- Properties of *receipt-freeness*, *coercion-resistance* and *voter-verifiability* in voting procedures [2, 16, 32],
- *Fairness* in contract-signing protocols and non-repudiation protocols [19, 20],
- Existence of winning strategies in *general games* [29], as well as specific multi-player games such as Bridge [15, 17].

3 SCENARIOS

The tool includes the following verification scenarios:

(1) Existence of a winning strategy in the ancient story of **TianJi** [23],
(2) Ability of a team of "workers" to defeat a given castle in the **Castles** benchmark from [27],
(3) Existence of a winning strategy for the declarer in the card game of Bridge (**Bridge Endplay** [15]),
(4) Ability of a team of drones to visit a given number of locations (the **Drones** benchmark [18]),
(5) A variant of coercion-resistance in a simple voting protocol (**Simple Voting** [15]).

4 FORMAL BACKGROUND

Models. The main part of the input is given by an *imperfect information concurrent game structure* [1, 31], i.e., a labeled multi-agent transition system with the transitions labeled by synchronous actions from all the agents in the system. The knowledge of each agent is represented by its epistemic indistinguishability relation. An example model is shown in Figure 1.

Strategies. A strategy is a conditional plan that specifies what the agent is 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 (formally, abstraction classes of its indistinguishability relations) 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.

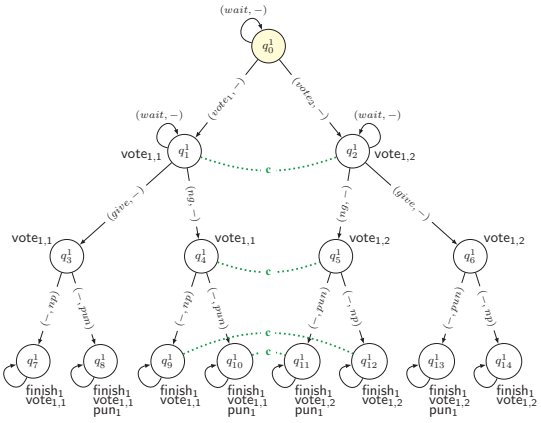

Figure 1: Simple voting model

Formulas. Given a model M and a state q in the model, the formula $\langle\langle A\rangle\rangle\gamma$ holds in (M, q) iff there exists a strategy for A that makes γ true on all the outcome paths starting from any state indistinguishable from q. For more details, we refer the reader to [1, 31].

5 TECHNOLOGY

STV does *explicit-state model checking*. That is, the states and transitions of the model are represented explicitly in the memory of the tool. We have implemented model generators for the scenarios presented in Section 3; the user sets the values of the scaling parameters (e.g., the number of drones and their initial level of energy), and the corresponding model is generated. After that, two approaches to model checking can be selected: *fixpoint approximation* and *dominance-based strategy search*.

Approximate fixpoint verification [15, 17]. The first approach is based on computing *fixpoint approximations* of the verified formula. Two formulas are produced for $\langle\langle A\rangle\rangle\gamma$:

- The *lower approximation* $tr_L(\langle\langle A\rangle\rangle\gamma)$ is a fixpoint expression in an extension of alternating epistemic μ-calculus [6] such that, if $tr_L(\langle\langle A\rangle\rangle\gamma)$ holds, then $\langle\langle A\rangle\rangle\gamma$ must hold as well;
- The *upper approximation* $tr_U(\langle\langle A\rangle\rangle\gamma)$ asks for perfect information strategies instead of imperfect information ones in the semantics of $\langle\langle A\rangle\rangle$. Thus, whenever $tr_U(\langle\langle A\rangle\rangle\gamma)$ returns *false*, $\langle\langle A\rangle\rangle\gamma$ must be false, too.

Depth-first search with removal of dominated strategies [21]. The second technique is based on a novel notion of strategic dominance, applied in an incremental, DFS-based search for a winning strategy. We say that partial strategy σ_A dominates σ'_A w.r.t. the context σ^C_A iff: (i) σ_A and σ'_A share the same set of input states, and (ii) for each input state q, the set of possible output states of σ_A is a subset of those for σ'_A. In other words, σ_A is "tighter" than σ'_A, and induces a smaller set of outcome paths.

The algorithm, called DominoDFS, attempts to expand the context strategy that contains the initial state by exploring its frontier. For each state at the current frontier, DominoDFS collects all the available one-step strategies, and then removes the dominated ones. Besides the basic version, we have implemented several heuristics that determine the order of the search.

Config.	DominoDFS	Approx.	Optimized approx.	MCMAS
(1, 1)	0.0006	0.0008	< 0.0001	0.12
(2, 2)	0.01	0.01	< 0.0001	8712
(3, 3)	0.8	0.8	0.06	timeout
(4, 4)	160	384	5.5	timeout
(5, 5)	1373	8951	39	timeout
(5, 5)	memout	memout	138	timeout
(6, 6)	memout	memout	4524	timeout

Table 1: Performance results for Bridge Endplay

Config.	DominoDFS	MCMAS	SMC
(1, 1, 1)	0.3	65	63
(2, 1, 1)	1.5	12898	184
(3, 1, 1)	25	timeout	6731
(2, 2, 1)	25	timeout	4923
(2, 2, 2)	160	timeout	timeout
(3, 2, 2)	2688	timeout	timeout
(3, 3, 2)	timeout	timeout	timeout

Table 2: Performance results for Castles

Implementation and evaluation. **STV** is implemented in Python 3. The algorithms have been evaluated on several benchmarks, with very promising results [15, 17, 21]. We used the state of the art model checker **MCMAS** [23] and the experimental tool **SMC** [27] as reference points. The results for Bridge endplay and Castles are shown in Tables 1 and 2, with the irrelevant columns omitted from the tables (fixpoint approximation is not applicable to Castles, and Bridge Endplay cannot be correctly encoded in **SMC**).

6 USAGE

The current version of **STV** (which can be found here) allows to:

- Select a class of predefined parameterised models and a predefined formula for verification (cf. Section 3),
- Set the values of the parameters that control scalability,
- Generate and display the explicit state-transition graph,
- Run the fixpoint approximation algorithm (lower and upper approximation),
- Run the dominance-based verification (DominoDFS),
- Display the verification result (truth value of the formula in the initial state of the model, states in the model where the formula holds, and possibly also the winning strategy that has been found).

7 CONCLUSIONS

Model checking strategic abilities under imperfect information is notoriously hard. Currently, no tools exist that would handle even toy examples in a satisfactory way. **STV** is our first step towards practical verification of such properties. We believe it is worth sharing with the MAS community even in this preliminary form.

REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49 (2002), 672–713. https://doi.org/10.1145/585265.585270

[2] F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A.V. Jones. 2017. Bisimulations for Verification of Strategic Abilities with Application to ThreeBallot Voting Protocol. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS, 1286–1295.

[3] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. 2017. Strategy logic with imperfect information. In *Proceedings of LICS*. 1–12. https://doi.org/10.1109/LICS.2017.8005136

[4] R. E. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers* 35, 8 (1986), 677–691.

[5] N. Bulling, J. Dix, and W. Jamroga. 2010. Model Checking Logics of Strategic Ability: Complexity. In *Specification and Verification of Multi-Agent Systems*, M. Dastani, K. Hindriks, and J.-J. Meyer (Eds.). Springer, 125–159.

[6] N. Bulling and W. Jamroga. 2011. Alternating Epistemic Mu-Calculus. In *Proceedings of IJCAI-11*. 109–114.

[7] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. 1990. Symbolic Model Checking: 10^20 States and Beyond. In *Proc. of 4th Ann. IEEE Symp. on Logic in Computer Science (LICS)*. IEEE Computer Society, 428–439.

[8] S. Busard, C. Pecheur, H. Qu, and F. Raimondi. 2015. Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. *Information and Computation* 242 (2015), 128–156. https://doi.org/10.1016/j.ic.2015.03.014

[9] P. Cermak, A. Lomuscio, F. Mogavero, and A. Murano. 2014. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *Proc. of Computer Aided Verification (CAV) (Lecture Notes in Computer Science)*, Vol. 8559. Springer, 525–532.

[10] P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. 2003. Verics: A Tool for Verifying Timed Automata and Estelle Specifications. In *Proceedings of the of the 9th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03)*. Lecture Notes in Computer Science, Vol. 2619. Springer, 278–283.

[11] C. Dima, B. Maubert, and S. Pinchinat. 2015. Relating Paths in Transition Systems: The Fall of the Modal Mu-Calculus. In *Proceedings of Mathematical Foundations of Computer Science (MFCS) (Lecture Notes in Computer Science)*, Vol. 9234. Springer, 179–191. https://doi.org/10.1007/978-3-662-48057-1_14

[12] L. Doyen and J.-F. Raskin. 2011. Games with Imperfect Information: Theory and Algorithms. In *Lecture Notes in Game Theory for Computer Scientists*. Cambridge University Press, 185–212.

[13] P. Gammie and R. van der Meyden. 2004. MCK: Model Checking the Logic of Knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04) (LNCS)*, Vol. 3114. Springer-Verlag, 479–483.

[14] W. Jamroga and J. Dix. 2006. Model Checking ATL$_{ir}$ is Indeed Δ_2^P-complete. In *Proceedings of EUMAS (CEUR Workshop Proceedings)*, Vol. 223.

[15] W. Jamroga, M. Knapik, and D. Kurpiewski. 2017. Fixpoint Approximation of Strategic Abilities under Imperfect Information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS, 1241–1249.

[16] W. Jamroga, M. Knapik, and D. Kurpiewski. 2018. Model Checking the SELENE E-Voting Protocol in Multi-Agent Logics. In *Proceedings of the 3rd International Joint Conference on Electronic Voting (E-VOTE-ID) (Lecture Notes in Computer Science)*, Vol. 11143. Springer, 100–116.

[17] W. Jamroga, M. Knapik, D. Kurpiewski, and Łukasz Mikulski. 2019. Approximate Verification of Strategic Abilities under Imperfect Information. *Artificial Intelligence* 277 (2019). https://doi.org/10.1016/j.artint.2019.103172

[18] W. Jamroga, B. Konikowska, W. Penczek, and D. Kurpiewski. 2019. Multi-Valued Verification of Strategic Ability. (2019). In preparation.

[19] W. Jamroga, S. Mauw, and M. Melissen. 2012. Fairness in Non-repudiation Protocols. In *Proceedings of STM'11 (Lecture Notes in Computer Science)*, Vol. 7170. 122–139.

[20] S. Kremer and J.-F. Raskin. 2003. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security* 11, 3 (2003). https://doi.org/10.1007/3-540-44685-0_37

[21] D. Kurpiewski, M. Knapik, and W. Jamroga. 2019. On Domination and Control in Strategic Ability. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. 197–205. http://dl.acm.org/citation.cfm?id=3331693

[22] D. Kurpiewski and D. Marmsoler. 2019. Strategic logics for collaborative embedded systems. (01 Dec 2019), 201–212 pages. https://doi.org/10.1007/s00450-019-00424-7

[23] A. Lomuscio, H. Qu, and F. Raimondi. 2015. MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems. *International Journal on Software Tools for Technology Transfer* (2015). https://doi.org/10.1007/s10009-015-0378-x Availabe online.

[24] A. Lomuscio and F. Raimondi. 2006. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 161–168. https://doi.org/10.1145/1160633.1160660

[25] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Transactions on Computational Logic* 15, 4 (2014), 1–42.

[26] F. Mogavero, A. Murano, and M.Y. Vardi. 2010. Reasoning About Strategies. In *Proceedings of FSTTCS*. 133–144.

[27] J. Pilecki, M.A. Bednarczyk, and W. Jamroga. 2017. SMC: Synthesis of Uniform Strategies and Verification of Strategic Ability for Multi-Agent Systems. *Journal of Logic and Computation* 27, 7 (2017), 1871–1895. https://doi.org/10.1093/logcom/exw032

[28] F. Raimondi and A. Lomuscio. 2007. Automatic Verification of Multi-agent Systems by Model Checking via Ordered Binary Decision Diagrams. *J. Applied Logic* 5, 2 (2007), 235–251.

[29] J. Ruan, W. van der Hoek, and M. Wooldridge. 2009. Verification of Games in the Game Description Language. *Journal of Logic and Computation* 19 (11 2009), 1127–1156. https://doi.org/10.1093/logcom/exp039

[30] B.-H. Schlingloff, H. Stubert, and W. Jamroga. 2016. Collaborative Embedded Systems - A Case Study. In *Proceedings of the 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC@CPSWeek)*. 17–22. https://doi.org/10.1109/EITEC.2016.7503691

[31] P. Y. Schobbens. 2004. Alternating-Time Logic with Imperfect Recall. *Electronic Notes in Theoretical Computer Science* 85, 2 (2004), 82–93.

[32] M. Tabatabaei, W. Jamroga, and Peter Y. A. Ryan. 2016. Expressing Receipt-Freeness and Coercion-Resistance in Logics of Strategic Ability: Preliminary Attempt. In *Proceedings of the 1st International Workshop on AI for Privacy and Security, PrAISe@ECAI 2016*. ACM, 1:1–1:8. https://doi.org/10.1145/2970030.2970039

[33] V. Yazdanpanah and M. Dastani. 2016. Quantified Group Responsibility in Multi-Agent Systems. In *Proceedings of WOA*. 44–49.

[34] V. Yazdanpanah, M. Dastani, N. Alechina, B. Logan, and W. Jamroga. 2019. Strategic Responsibility Under Imperfect Information. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. 592–600. http://dl.acm.org/citation.cfm?id=3331745