On Domination and Control in Strategic Ability (Extended Abstract)

Damian Kurpiewski[a] Michał Knapik[a] Wojciech Jamroga[a,b]

[a] *Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland*
[b] *Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg*

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. Many relevant properties of multi-agent systems refer to strategic abilities of agents and their groups. For example, functionality requirements can be often understood in terms of the user's ability to complete the selected tasks. Similarly, many security properties boil down to inability of the intruder to obtain his goals. *Logics of strategic ability* provide powerful tools to reason about such aspects of MAS [1, 8, 6]. A typical property to express is that *the group of agents A has a collective strategy to enforce temporal property φ, no matter what the other agents do*.

Verification of such properties, especially for strategies with imperfect information, is difficult for a number of reasons. In particular, incremental algorithms do not work, and the space of strategies is huge – usually larger than the state space by orders of magnitude. However, some strategies are better than others. Here, we propose and study a notion of strategic dominance that refers to the amount of control obtained by a strategy. Intuitively, those strategies are better which have a tighter control on the system dynamics.

The formal definitions and a detailed presentation of the results can be found in the original paper [3]. A prototype tool implementing our model checking algorithm is described in [4].

2 Comparing Partial Strategies

We propose and study a notion of domination that refers to the tightness of the strategy. Technically, this is defined by introducing a new concept of *input/output characteristic of a (partial) strategy*. The inputs of a strategy consist of all the states where the strategy is granted the full control over the execution of the system. To each of these entry points we assign the set of states where the strategy returns the control to the environment, i.e., the outputs. A new strategy is better than the original one if it assigns smaller outputs to the same inputs.

We prove that the notion of dominance based on the comparison of input/output characteristics is sound, i.e., a dominating strategy can achieve at least what the dominated one can. On the other hand, dominance does not necessarily lead to simpler strategies. We thus combine the theoretical concept with heuristics geared towards simplicity of strategies.

3 Model Checking with DominoDFS: Algorithm and Evaluation

We have used the new concept of dominance to design and implement an on-the-fly model checking algorithm that tries to synthesise a winning strategy. The main routine is based on depth-first search and

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

Conf.	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 1: Results for Bridge Endplay (left) and Castles (right). For the configurations marked with (*), tests were only run on a single handcrafted instance of the model due to timeout or memout

synthesis, starting from the initial state. The novelty of the approach consists in elimination of dominated partial strategies that are candidates for including in the final result. This substantially reduces the search space, as demonstrated by very promising experimental results.

The algorithm, called DominoDFS, has been implemented in Python 3. We used non-symbolic representations, i.e., the models are stored in memory explicitly. We compared the performance of DominoDFS to three existing tools: the state of the art tool MCMAS [5], an experimental model checker SMC [7], and a prototype implementation (in C++) of the fixpoint approximation algorithms of [2]. The experimental results for the benchmarks of Bridge Endplay [2] and Castles [7] are shown in Table 1. All the tests have been conducted on a laptop with an Intel Core i7-6700HQ CPU with dynamic clock speed of 2.60 GHz up to 3.50 GHz, 32 GB RAM, and 64bit Linux. The running times are given in seconds; the timeout was 4h.

The results show that DominoDFS significantly outperforms MCMAS and SMC. It also successfully competes with the basic implementation of fixpoint approximation. We also note that our new approach can handle models that do not submit to the fixpoint approximation scheme (i.e., Castles), as well as ones on which the output of SMC is faulty (i.e., Bridge Endplay).

References

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

[2] W. Jamroga, M. Knapik, and D. Kurpiewski. Fixpoint approximation of strategic abilities under imperfect information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1241–1249. IFAAMAS, 2017.

[3] D. Kurpiewski, M. Knapik, and W. Jamroga. On domination and control in strategic ability. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 197–205. IFAAMAS, 2019.

[4] D. Kurpiewski, M. Knapik, and W. Jamroga. 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.

[5] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 2015.

[6] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *Proceedings of FSTTCS*, pages 133–144, 2010.

[7] J. Pilecki, M.A. Bednarczyk, and W. Jamroga. SMC: Synthesis of uniform strategies and verification of strategic ability for multi-agent systems. *Journal of Logic and Computation*, 27(7):1871–1895, 2017.

[8] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.