S-IMITATOR: Strategy Synthesis for STCTL

Davide Catta[1], Adrien Lacroix[1,2], Wojciech Penczek[3], Laure Petrucci[1], and Teofil Sidoruk[3]

[1] LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord, Villetaneuse, France
[2] École Centrale de Nantes, France
[3] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. S-IMITATOR supports model checking and synthesis of memoryless imperfect information strategies for **STCTL**, interpreted over networks of parametric timed automata with asynchronous execution. While extending the verifier IMITATOR, S-IMITATOR is the first tool for strategy synthesis in this setting. Our experimental results show a substantial speedup over previous approaches.

1 Introduction

A multi-agent system models a computing system as the result of interactions between rational agents acting independently or cooperatively to achieve a goal. [21]. Studying multi-agent systems therefore means examining the strategic abilities of a group of agents [16,9]. In order to formally verify whether the executions produced by their interactions behave as expected, strategic logics were introduced, augmenting temporal logics with strategy operators [2,9,10,5,7]. They express that a group of agents has a strategy to enforce a given temporal property, regardless of the actions executed by the agents outside the group. Formal verification using strategic logics addresses two main problems [8], given a model of the system and a strategic formula, :

1. **Model checking:** determine whether the formula holds in the initial state of the model.
2. **Strategy synthesis:** explicitly construct a strategy that ensures the temporal property is satisfied.

A solution to the latter naturally yields a solution to the former.

Strategic Timed Computation Tree Logic (**STCTL**) [5] is a powerful logic designed to describe the behaviour of systems whose executions are subject to explicit time constraints. It offers a natural framework for capturing scenarios in which the timing of actions is essential for meeting the specification. In this paper, we focus on the synthesis of memoryless strategies for agents having imperfect information about the system. This assumption is essential: in many real-world scenarios, agents do not have full visibility of the global state, and their decisions must rely solely on their observation of the system. Working under imperfect information therefore captures a more realistic view of strategic reasoning [7].

Application Domains. Strategy synthesis is particularly valuable in safety-critical and time-sensitive multi-agent applications, such as industrial automation, avionics, and distributed control systems, where correctness and timely execution are paramount [17,19,20]. Attack-defence trees [14] representing relevant security scenarios can be easily translated to the multi-agent setting of our tool [6], and studied in a new, strategic context of opposing coalitions [4,13]. Socio-technical interactions, e.g. e-voting [15] emerge as a use case.

2 Theoretical Background

In this section, we introduce the relevant theoretical background.

Multi-agent systems. A continuous-time asynchronous multi-agent system (CAMAS) [5] models agents as components of a timed automata network. Clock constraints are enforced in transition guards and state invariants [1]. Agents interleave private actions independently, but must synchronise on shared ones. Their product (or CAMAS *model*) captures the system's global behaviour.

Strategic ability. Conditional plans defining choices to be made by agents are called joint *strategies*, and classified by state information (perfect I vs. imperfect i) and memory (perfect recall R vs. memoryless r) available to agents [18]. We focus on ir-strategies, i.e. decisions based solely on the current local state.

Logic. Strategic Timed **CTL** (**STCTL**) [5] extends **CTL** by adding time constraints and strategic operators. It is defined by the syntax:
$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle \gamma, \quad \gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \forall\gamma \,\mathtt{U}_I\, \gamma \mid \exists\gamma \,\mathtt{U}_I\, \gamma \mid \forall\gamma \,\mathtt{R}_I\, \gamma \mid \exists\gamma \,\mathtt{R}_I\, \gamma$
where p is an atomic proposition, A is a subset of agents, $\langle\langle A \rangle\rangle \gamma$ expresses that coalition A has a strategy to enforce the property γ, and intervals $I \subseteq \mathbb{R}_{0+}$ denote time constraints on the evaluation of temporal operators. The remaining **CTL** path quantifiers ($\forall, \exists$), temporal operators ($\mathtt{U}_I$, $\mathtt{R}_I$, with $\mathtt{F}_I, \mathtt{G}_I$ derived), and Boolean connectives ($\neg, \wedge$) are standard (see [5] for their formal semantics).

3 Architecture, Technology, and usage for synthesis

The implementation of S-IMITATOR builds on the OCaml model checker IMITATOR [3], which supports **TCTL** properties over networks of Parametric Timed Automata with asynchronous execution. Parameters with unknown values can appear in transition guards and state invariants. Global states are tuples of automata locations and constraints on parameters and clocks. In the strategic setting, automata are viewed as autonomous agents with individual behaviours that synchronise on shared actions. Transitions are labelled by actions chosen as part of a strategy. For **STCTL** model checking and strategy synthesis, global states include the current strategy. A successor is added only if its strategy matches the source state's strategy, or if none exists, updating the target state's strategy. The constructed state space only contains the executions consistent with the strategy. The **TCTL** property inside the strategic operator is checked on-the-fly, and the tool displays the valid strategies and associated parameters[4]. This process might not terminate but the results obtained are sound.

[4] Timing parameters synthesis is obtained here for free by extending IMITATOR.

4 Experimental Evaluation

We compare the efficiency of S-IMITATOR with the previous approach involving IMITATOR [5], on a benchmark of the voting example [5], though we additionally scale the size of the coalition. The formula $\langle\langle V_i \rangle\rangle \exists \mathsf{F}_{[0;8]} \mathsf{voted}_{i,1}$ specifies that voter V_i has a strategy to vote for the first candidate within 8 time units. In the experiments, we synthesise *all* strategies (by inputting the property with the `#synth` keyword). The results in Table 1 exhibit huge gains (up to 35x) compared to the initial approach of [5], where parameters were used to encode the strategies within the model. The improvements stem from three structural effects: (i) merging of equivalent execution prefixes, (ii) early pruning of incompatible strategic branches, and (iii) termination upon detection of a suitable strategy. These mechanisms substantially reduce the explored symbolic state space.

		IMITATOR				S-IMITATOR			
—A—	v	c=1	c=2	c=3	c=4	c=1	c=2	c=3	c=4
1	1	0.01	0.01	0.02	0.03	0.01	0.01	0.01	0.01
1	2	0.06	0.14	0.38	0.70	0.02	0.03	0.04	0.05
1	3	0.67	2.15	6.49	16.5	0.10	0.21	0.44	0.81
1	4	6.03	26.4	78.0	timeout	0.76	2.74	7.95	21.3
1	5	45.4	timeout			9.51	90.8	timeout	
2	2	0.09	0.25	0.62	1.38	0.02	0.03	0.04	0.05
2	3	0.92	3.55	10.0	27.5	0.12	0.27	0.53	1.02
2	4	10.4	53.9	timeout		1.01	3.75	12.0	34.0
2	5	98.1	timeout			15.5	timeout		
3	3	1.21	5.16	17.2	47.2	0.12	0.29	0.65	1.33
3	4	14.4	67.3	timeout		1.18	4.72	16.4	53.7
3	5	timeout				20.3	timeout		

Table 1. Synthesising all strategies in the voting benchmark with v voters, c candidates, and agent coalition of size $|A|$.

Additionally, two examples include extra features: the treasure hunters [13] support both timing parameter synthesis and strategy computation, while a variant of the conference scenario [11] demonstrates a joint strategy for multiple agents.

5 Conclusions and Future Work

S-IMITATOR provides an integrated environment for synthesis of memoryless strategies with imperfect information, as a push-button procedure. The only input needed from the user are the model and the property. Compared to other approaches, it does not require additional expertise. This tool constitutes a major step forward in strategy synthesis for real-time multi-agent systems.

An important direction for further extending S-IMITATOR is adding support for other strategy semantics, e.g. counting [12], timed, or an intermediate "partial view" between imperfect and perfect information, with strategic choices based on the local states of a *subset* of agents (not necessarily from the coalition).

References

1. R. Alur and D. Dill. The theory of timed automata. In *Real-Time: Theory in Practice*, pages 45–73. Springer, 1992.
2. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
3. É. André. Imitator 3: Synthesis of timing parameters beyond decidability. In *Proc. of CAV 2021*, pages 552–565. Springer, 2021.
4. J. Arias, C. Budde, W. Penczek, L. Petrucci, T. Sidoruk, and M. Stoelinga. Hackers vs. Security: Attack-Defence Trees as Asynchronous Multi-agent Systems. In *Proc. of ICFEM 2020*, pages 3–19. Springer, 2020.
5. J. Arias, W. Jamroga, W. Penczek, L. Petrucci, and T. Sidoruk. Strategic (Timed) Computation Tree Logic. In *Proc. of AAMAS'23*, pages 382–390. ACM, 2023.
6. J. Arias, W. Penczek, L. Petrucci, and T. Sidoruk. ADT2AMAS: Managing Agents in Attack-Defence Scenarios. In *Proc. of AAMAS '21*, pages 1749–1751. ACM, 2021.
7. R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. Strategy logic with imperfect information. In *Proc. of LICS 2017*, pages 1–12. IEEE Computer Society, 2017.
8. 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.
9. W. Jamroga. *Logical Methods for Specification and Verification of Multi-Agent Systems*. ICS PAS Publishing House, 2015.
10. W. Jamroga. Model checking strategic ability - why, what, and especially: How? In *Proc. of TIME 2018*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
11. W. Jamroga, W. Penczek, and T. Sidoruk. Strategic abilities of asynchronous agents: Semantic side effects and how to tame them. In *Proc. of KR 2021*, pages 368–378, 2021.
12. M. Knapik, É. André, L. Petrucci, W. Jamroga, and W. Penczek. Timed ATL: forget memory, just count. *J. Artif. Intell. Res.*, 66:197–223, 2019.
13. M. Knapik, W. Penczek, L. Petrucci, and T. Sidoruk. Squeezing State Spaces of (Attack-Defence) Trees. In *Proc. of ICECCS 2019*, pages 71–80. IEEE, 2019.
14. B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Foundations of Attack-Defense Trees. In *Proc. of FAST 2010*, pages 80–95. Springer, 2011.
15. D. Kurpiewski, W. Jamroga, Ł. Maśko, Ł. Mikulski, W. Pazderski, W. Penczek, and T. Sidoruk. Verification of Multi-Agent Properties in Electronic Voting: A Case Study. In *Proc. of AiML 2022*, pages 531–556. College Publications, 2022.
16. A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *Proc. of AAMAS'06*, pages 161–168, 2006.
17. C. Rochange. Parallel real-time tasks, as viewed by WCET analysis and task scheduling approaches. In *Proc. of WCET 2016*, pages 1–11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
18. P.-Y. Schobbens. Alternating-time Logic with Imperfect Recall. In *Proc. of LCMAS 2003*, pages 1–12. Elsevier, 2004.
19. A. Singh. Cutting-plane algorithms for preemptive uniprocessor scheduling problems. *Real Time Syst.*, 60(1):24–73, 2024.
20. M. Wolf. Chapter 9 - automotive and aerospace systems. In *Computers as Components*, pages 437–452. Morgan Kaufmann, 2023.
21. M. J. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2009.