SMT4SMTL: a Tool for SMT-Based Satisfiability Checking of SMTL

Demonstration Track

Artur Niewiadomski
University of Siedlce
Siedlce, Poland
artur.niewiadomski@uws.edu.pl

Maciej Nazarczuk
University of Siedlce
Siedlce, Poland
maciej.nazarczuk@uws.edu.pl

Mateusz Przychodzki
University of Siedlce
Siedlce, Poland
mateusz.przychodzki@uws.edu.pl

Magdalena Kacprzak
Bialystok University of Technology
Bialystok, Poland
m.kacprzak@pb.edu.pl

Wojciech Penczek
Institute of Computer Science, PAS
Warsaw, Poland
w.penczek@ipipan.waw.pl

Andrzej Zbrzezny
Jan Dlugosz University
in Czestochowa, Poland
a.zbrzezny@ujd.edu.pl

ABSTRACT

We present SMT4SMTL - the first tool for deciding the bounded satisfiability of Metric Temporal Logic (MTL) and the existential fragment of Strategic Metric Temporal Logic (SMTL), interpreted over timed multi-agent systems represented by networks of timed automata. The tool combines Satisfiability Modulo Theories (SMT) techniques and Parametric Bounded Model Checking algorithms.

KEYWORDS

Strategic MTL; Bounded Satisfiability; SMT

ACM Reference Format:
Artur Niewiadomski, Maciej Nazarczuk, Mateusz Przychodzki, Magdalena Kacprzak, Wojciech Penczek, and Andrzej Zbrzezny. 2024. SMT4SMTL: a Tool for SMT-Based Satisfiability Checking of SMTL: Demonstration Track. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 INTRODUCTION AND CHALLENGES

The paper presents the tool SMT4SMTL for solving the *bounded satisfiability problem* for the existential fragment of *Strategic Metric Temporal Logic* (SMTL) [21] and MTL [6, 24]. This is a problem to decide whether an SMTL formula is satisfiable on a *timed multi-agent system* under some initial restrictions. SMT4SMTL implements a new method of SAT checking [21], which combines Satisfiability Modulo Theories (SMT) techniques and Parametric Bounded Model Checking algorithms. The bounded approach can be used only for the existential fragment of SMTL. Our method consists in synthesising a model as the product of agents represented by a Parametric Network of Timed Automata (PNTA), for an SMTL formula expressing the property. Parameters are used in guards, invariants, and to specify transitions and actions. Given an additional knowledge about the system's structure, some parameters can be replaced with fixed values. We encode the SMTL formula and the runs of PNTA unfolded to a depth k, as an SMT problem instance, which is then checked for satisfiability by an SMT-solver. If the answer is SAT, all parameter values from a model are returned by the SMT-solver. Otherwise, the unfolding depth is increased.

The main challenge in solving the SAT problem of MTL and SMTL is its high complexity. The SAT problem for MTL, so for SMTL, is undecidable. In order to regain decidability of MTL, certain semantic and syntactic restrictions are introduced [36]. Semantic restrictions include adopting an integer-time model [6, 14, 16] or a bounded-variation dense-time model [42] for which SAT is decidable. Syntactic restrictions concern: punctuality or non-singularity [5], boundedness and safety [9, 34, 35]. Then, the SAT problem becomes decidable and is EXPSPACE-complete. To the best of our knowledge, SMT4SMTL is the only tool solving the SAT problem for SMTL. However, there are tools for solving the SAT problem for untimed strategic temporal logics [19, 20, 33].

2 APPLICATION DOMAIN

In recent years, significant efforts have been dedicated to verifying strategic properties in Multi-Agent Systems (MAS) [2, 17, 18, 22, 23, 37, 41] expressed using formalisms such as *Alternating-time Temporal Logic* ATL and ATL* [7], as well as its generalization *Strategy Logic* (SL) [31], also with several restrictions [1, 8, 28–30], This research is supported by the tools like MOCHA [3], MCMAS [27], STV [25, 26], or MCMAS-SLK [10–12]. The primary emphasis lies on strategy synthesis, closely intertwined with model checking [11, 12]. Model synthesis, in turn, involves the automatic construction of a model for a given formula, thereby verifying the existence of such a model. SMTL finds application in specifying strategy-oriented behavior and troubleshooting in real-time systems, which are often complex and challenging to design, implement, and test, requiring specialized skills and expertise. These systems are engineered for real-time operation, demanding guaranteed responses within specified periods or meeting particular deadlines [38, 39]. For instance, controllers in airplanes, cars, or industrial plants are expected to complete tasks within reliable time constraints [43]. Additional potential applications encompass real-time communication and real-time strategy games.

3 THEORETICAL BACKGROUND

Strategic Metric Temporal Logic (SMTL) [21] extends Metric Temporal Logic (MTL) by strategy operators $\langle\langle A\rangle\rangle\exists$ and $\langle\langle A\rangle\rangle\forall$ (preceding MTL formulae) for specifying existential and universal (resp.) strategic properties. Let $p \in \mathcal{AP}$ be an atomic proposition, *Id* be

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

a set agents, and $\mathcal{J}$ the set of all the intervals in $\mathbb{R}_+$ of the form $[a,a]$, (a,b), $[a,b)$, $(a,b]$, $[a,b]$, (a,∞), or $[a,\infty)$, where $a,b \in \mathbb{N}$ and $a < b$, and let $\mathrm{I} \in \mathcal{J}$. The syntax of SMTL (MTL) is defined by the formulae ϕ (ψ, resp.) as follows:
$\phi := p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\langle \mathrm{A} \rangle\rangle \exists \psi \mid \langle\langle \mathrm{A} \rangle\rangle \forall \psi$,
$\psi := p \mid \neg p \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \mathrm{U_I} \psi \mid \mathrm{G_I}\psi$, where $A \subseteq Id$.
The operators $\mathrm{U_I}$ and $\mathrm{G_I}$ are read as "until in the interval I" and "always in the interval I". The operator $\mathrm{F_I}$ is defined in the standard way: $\mathrm{F_I}\psi = \mathbf{true}\,\mathrm{U_I}\,\psi$, where $\mathbf{true} := p \vee \neg p$, for some $p \in \mathcal{AP}$. Intuitively, $\langle\langle \mathrm{A} \rangle\rangle \exists \psi$ means that the agents of A have a collective strategy s.t. it is possible to ensure ψ, while $\langle\langle \mathrm{A} \rangle\rangle \forall \psi$ means that the agents of A have a strategy to inevitably ensure ψ. The fragment of SMTL is called *existential* if it does not contain the subformulas $\langle\langle \mathrm{A} \rangle\rangle \forall \psi$ and the negation is applied to the propositions only.

SMTL is interpreted over concrete models of Continuous MAS (CMAS), where each agent is represented by a timed automaton [4] with asynchronous, strongly monotonic semantics and continuous time. Thus, we assume that CMAS consists of n agents, each assigned a set of *local actions*, a set of *local states*, an *initial local state*, a set of *local clocks*, a *local transition relation* defining possible changes of local states (clocks can be reset), a *local protocol* that assigns a non-empty set of available actions to each state, and a *state invariant function* that assigns clock constraints to the local states. The global states are tuples of the local states, and the global transition relation is defined by the asynchronous composition of the local transition relations of all agents. A *strategy* of agent i is a conditional plan that specifies what i is going to do in any situation. We focus on *memoryless imperfect information* strategies for each agent i, which intuitively, assigns a local action to each of its local states. For more details of the logic and the encoding see [21].

The problem we are addressing is the determination of the satisfiability of an existential SMTL formula ϕ, i.e., SMT4SMTL checks whether there is a model M for ϕ. This is achieved by defining a PNTA with meta-parameters specifying the number of timed automata, as well as the number of their local states and transitions. This network and ϕ are encoded in SMT using Boolean, Integer, and Real variables. Finding a model involves determining values for these variables. The algorithm terminates when either a model satisfying ϕ is discovered or when the memory/time limit is reached.

4 ARCHITECTURE AND TECHNOLOGY

There are three main modules of SMT4SMTL: GUI, BMC [21], and Z3 SMT-solver[32]. GUI is a user friendly, interactive web client implemented in TypeScript on the top of SvelteKit [15] and Cytoscape [13] libraries. It allows to edit graphically a formula as well as PNTA and start the computations on the server side. The BMC module, implemented in C++, encodes the problem using smtlibv2 standard, which is then checked for satisfiability by the Z3 SMT-solver. When the computations are complete, the GUI visualizes the results (see Fig. 1). For more details the reader is referred to the tool website: https://smtl.ii.uws.edu.pl/.

5 EXPERIMENTAL EVALUATION

Tab. 1 displays an evaluation of SMT4SMTL performance using the timed version of Dining Philosophers problem of [21] and the formula $\alpha = \langle\langle Lck \rangle\rangle\ \exists\ (F_{[1,E_2)}(\bigwedge_{j\in odd_p} Eating_j) \wedge (\bigwedge_{j\in odd_p}(F_{[0,\infty)} Hungry_j \wedge F_{[0,\infty)} Waiting_j \wedge F_{[0,\infty)} Released_j)))$ - the lackey has a strategy s.t. it is possible that all odd philosophers but the last one meet at the table at some point of time between 1 and E_2 and eat, and all of the crucial philosophers' locations must be reachable, where $odd_p = \{j \mid 1 \leq j < p \wedge j\ mod\ 2 = 1\}$. The second part of Tab. 1 reports the results for the crossroad system, inspired by [40]. The formula $\beta = \bigwedge_{i=1} \langle\langle Lck \rangle\rangle\ \exists\ (G_{[0,\infty)}(noCol \wedge F_{[0,5)}(in_i \wedge F_{[3,8)}(leave_i \wedge F_{[6,11)} out_i))))$, where $noCol = \bigwedge_{i=1..n-1,j=i+1..n}(\neg leave_i \vee \neg leave_j)$, means that each car has a strategy s.t. it can drive trough the crossroad without colliding with any other car. The meaning of the table columns, from left to right, is: the parameter variant, the numbers of philosophers/cars, agents, the length of a shortest path satisfying the formula, time (in sec.) consumed by BMC / SMT-solver Z3, and maximal memory usage (in GB). There are two parameter variants: bSAT - everything is a parameter, no constraints imposed; Synth - a controller synthesis: all agents but the controller are fully specified. The experiments were performed on a server equipped with an Intel Xeon Gold 6234 3.30GHz CPU and 192GB RAM running Ubuntu Linux 22.04.3 LTS and Z3-solver v4.8.12. While not comprehensive, the results show the potential of the method, especially for some classes of SMTL formulae. The time of synthesis increases together with the number of agents, their states and actions, and complexity of the formulae. Clearly, the run-time decreases if the user provides also a partial specification of the system to be synthesised.

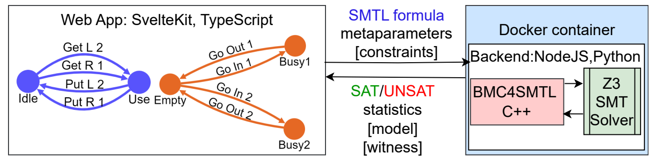

Figure 1: SMT4SMTL architecture.

		α					β			
Var	**p**	**n**	**k**	**Time**	**Mem**	**n**	**k**	**Time**	**Mem**
bSAT	2	5	8	0.6 / 21.2	0.2	3	6	0.1 / 1.5	0.05
	3	7	8	2.8 / 58.5	0.3	4	6	0.2 / 6.0	0.09
	4	9	8	8.6 / 1416	1.6	5	6	0.4 / 17	0.17
	5	11	8	23 / 3420	3.4	6	6	0.8 / 99	0.38
Synth	2	5	12	1.3 / 3.8	0.1	3	8	0.3 / 1.5	0.04
	3	7	12	6.0 / 12.6	0.1	4	8	0.6 / 2.1	0.05
	4	9	24	78 / 298	0.5	5	8	1.1 / 3.3	0.07
	5	11	24	202 / 491	0.9	6	8	1.9 / 5.2	0.10

Table 1: Experimental results.

6 CONCLUSIONS

Our tool implements a novel technique for bounded satisfiability checking of a fragment of SMTL. This marks a breakthrough in the field, as SMT4SMTL stands out as the first tool capable of synthesizing systems specified within a fragment of SMTL. The method can also be applied to partially specified systems. The experiments conducted demonstrate a high potential for this approach.

ACKNOWLEDGMENTS

M. Kacprzak was supported by the Bialystok University of Technology within the grant WZ/WI-IIT/2/2022. W. Penczek was supported by NCBR Poland under the PolLux/FNR-CORE project SpaceVote (POLLUX-XI/14/SpaceVote/2023) and PHC Polonium project MoCcA.

REFERENCES

[1] Erman Acar, Massimo Benerecetti, and Fabio Mogavero. 2019. Satisfiability in Strategy Logic Can Be Easier than Model Checking. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (2019), 2638–2645.

[2] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. 2007. Alternating-time Temporal Logics with Irrevocable Strategies. In *Proceedings of the TARK Conference*, Dov Samet (Ed.). 15–24.

[3] Rajeev Alur, Luca de Alfaro, Radu Grosu, Thomas A. Henzinger, M. Kang, Christoph M. Kirsch, Rupak Majumdar, Freddy Y. C. Mang, and Bow-Yaw Wang. 2001. JMOCHA: A Model Checking Tool that Exploits Design Structure. In *Proc. of the ICSE Conference*. IEEE Computer Society, 835–836.

[4] Rajeev Alur and David Dill. 1992. The Theory of Timed Automata. In *Real-Time: Theory in Practice*, J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg (Eds.). Springer Berlin Heidelberg, 45–73.

[5] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. 1996. The Benefits of Relaxing Punctuality. *J. ACM* 43, 1 (1996), 116–146.

[6] Rajeev Alur and Thomas A. Henzinger. 1993. Real-Time Logics: Complexity and Expressiveness. *Information and Computation* 104, 1 (1993), 35–77.

[7] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time Temporal Logic. *J. ACM* 49(5) (2002), 672–713.

[8] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2017. Verification of Broadcasting Multi-Agent Systems against an Epistemic Strategy Logic. In *Proc. of the IJCAI Conference*, Carles Sierra (Ed.). 91–97.

[9] Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. 2008. On Expressiveness and Complexity in Real-Time Model Checking. In *Automata, Languages and Programming, Proc. of the ICALP Conference (LNCS, Vol. 5126)*. Springer, 124–135.

[10] Petr Cermák, Alessio Lomuscio, Fabio Mogavero, and Aniello Murano. 2014. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *Computer Aided Verification - Proc. of the CAV Conference (LNCS, Vol. 8559)*, Armin Biere and Roderick Bloem (Eds.). Springer, 525–532.

[11] Petr Cermák, Alessio Lomuscio, Fabio Mogavero, and Aniello Murano. 2018. Practical Verification of Multi-Agent Systems against SLK Specifications. *Inf. Comput.* 261, Part (2018), 588–614.

[12] Petr Cermák, Alessio Lomuscio, and Aniello Murano. 2015. Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications. In *Proc. of the AAAI Conference on Artificial Intelligence*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 2038–2044.

[13] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. 2015. Cytoscape.js: a Graph Theory Library for Visualisation and Analysis. *Bioinformatics* 32, 2 (2015), 309–311.

[14] Carlo A. Furia and Paola Spoletini. 2008. Tomorrow and All our Yesterdays: MTL Satisfiability over the Integers. In *Theoretical Aspects of Computing - ICTAC 2008 (LNCS, Vol. 5160)*. Springer, 126–140.

[15] Rich Harris. 2024. Svelte Kit: The Fastest Way to Build Svelte Apps. https://kit.svelte.dev/.

[16] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. 1992. What Good are Digital Clocks?. In *Automata, Languages and Programming*, W. Kuich (Ed.). Springer Berlin Heidelberg, 545–558.

[17] Wojciech Jamroga and Wiebe van der Hoek. 2004. Agents that Know How to Play. *Fundamenta Informaticae* 63, 2-3 (2004), 185–219.

[18] Magdalena Kacprzak, Alessio Lomuscio, and Wojciech Penczek. 2004. Verification of Multi-Agent Systems via Unbounded Model Checking. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe (Eds.), Vol. II. ACM, 638–645.

[19] Magdalena Kacprzak, Artur Niewiadomski, and Wojciech Penczek. 2020. SAT-Based ATL Satisfiability Checking. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'20)*, D. Calvanese, E. Erdem, and M. Thielscher (Eds.). 539–549.

[20] Magdalena Kacprzak, Artur Niewiadomski, and Wojciech Penczek. 2021. Satisfiability Checking of Strategy Logic with Simple Goals. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning, (KR'21)*, M. Bienvenu, G. Lakemeyer, and E. Erdem (Eds.). 400–410.

[21] Magdalena Kacprzak, Artur Niewiadomski, Wojciech Penczek, and Andrzej Zbrzezny. 2023. SMT-Based Satisfiability Checking of Strategic Metric Temporal Logic. In *Proc. of the ECAI Conference (Frontiers in Artificial Intelligence and Applications, Vol. 372)*. IOS Press, 1180–1189.

[22] Magdalena Kacprzak and Wojciech Penczek. 2004. Unbounded Model Checking for Alternating-time Temporal Logic. In *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe (Eds.), Vol. II. ACM, 646–653.

[23] Magdalena Kacprzak and Wojciech Penczek. 2005. Fully Symbolic Unbounded Model Checking for Alternating-time Temporal Logic. *Auton. Agents Multi Agent Syst.* 11, 1 (2005), 69–89.

[24] Ron Koymans. 1990. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Syst.* 2, 4 (1990), 255–299.

[25] Damian Kurpiewski, Wojciech Jamroga, and Michal Knapik. 2019. STV: Model Checking for Strategies under Imperfect Information. In *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'19)*. 2372–2374.

[26] Damian Kurpiewski, Witold Pazderski, Wojciech Jamroga, and Yan Kim. 2021. STV+Reductions: Towards Practical Verification of Strategic Ability Using Model Reductions. In *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'21)*. ACM, 1770–1772.

[27] Alessio Lomuscio and Franco Raimondi. 2006. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*. 161–168.

[28] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comput. Log.* 15, 4 (2014), 34:1–34:47.

[29] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. 2017. Reasoning about Strategies: on the Satisfiability Problem. *Log. Methods Comput. Sci.* 13, 1 (2017).

[30] Fabio Mogavero, Aniello Murano, and Luigi Sauro. 2013. On the Boundary of Behavioral Strategies. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. IEEE Computer Society, 263–272.

[31] Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. 2010. Reasoning About Strategies. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010 (LIPIcs, Vol. 8)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 133–144.

[32] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: an Efficient SMT Solver. In *Proc. of the TACAS Conference (LNCS, Vol. 4963)*. Springer-Verlag, 337–340.

[33] Artur Niewiadomski, Magdalena Kacprzak, Damian Kurpiewski, Michal Knapik, Wojciech Penczek, and Wojciech Jamroga. 2020. MsATL: A Tool for SAT- Based ATL Satisfiability Checking. In *Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'20*. International Foundation for Autonomous Agents and Multiagent Systems, 2111–2113.

[34] Joël Ouaknine and James Worrell. 2006. Safety Metric Temporal Logic Is Fully Decidable. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin Heidelberg, 411–425.

[35] Joël Ouaknine and James Worrell. 2007. On the Decidability and Complexity of Metric Temporal Logic over Finite Words. *Logical Methods in Computer Science* Volume 3, Issue 1 (2007).

[36] Joël Ouaknine and James Worrell. 2008. Some Recent Results in Metric Temporal Logic. In *Formal Modeling and Analysis of Timed Systems*. Springer Berlin Heidelberg, 1–13.

[37] Marc Pauly. 2002. A Modal Logic for Coalitional Power in Games. *J. Log. Comput.* 12, 1 (2002), 149–166.

[38] Christine Rochange. 2016. Parallel Real-Time Tasks, as Viewed by WCET Analysis and Task Scheduling Approaches. In *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016) (Open Access Series in Informatics (OASIcs), Vol. 55)*, Martin Schoeberl (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 11:1–11:11.

[39] Abhishek Singh. 2023. Cutting-plane Algorithms for Preemptive Uniprocessor Scheduling Problems. *Real-Time Systems* (2023), 1–50.

[40] Masoud Tabatabaei, Wojciech Jamroga, and Peter Y. A. Ryan. 2016. Expressing Receipt-Freeness and Coercion-Resistance in Logics of Strategic Ability: Preliminary Attempt. In *Proc. of the 1st International Workshop on AI for Privacy and Security, PrAISe@ECAI 2016*. ACM, 1:1–1:8.

[41] Dirk Walther, Wiebe van der Hoek, and Michael J. Wooldridge. 2007. Alternating-time Temporal Logic with Explicit Strategies. In *Proc. of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007)*, Dov Samet (Ed.). 269–278.

[42] Thomas Wilke. 1994. Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In *Proceedings of the Third International Symposium Organized Jointly with the Working Group Provably Correct Systems on Formal Techniques in Real-Time and Fault- Tolerant Systems (ProCoS)*. Springer-Verlag, 694–715.

[43] Marilyn Wolf. 2023. Chapter 5 - Program Design and Analysis. In *Computers as Components* (fifth edition ed.), Marilyn Wolf (Ed.). Morgan Kaufmann, 219–319.