LTL Modulo Theories over Finite Traces: Modeling, Verification, Open Questions

Alessandro Gianola[1], Nicola Gigante[1]

[1]Free University of Bozen-Bolzano, Italy

Abstract

Traditional approaches to modeling and verification of systems based on model checking of linear-time properties are usually *propositional* in nature (*e.g.*, LTL). However, many real-world applications, such as business process management and knowledge-base-driven systems, handle data that intrinsically requires first-order formalisms. Recently, the LTLf Modulo Theory ($\mathsf{LTLf}^{\mathsf{MT}}$) has been introduced to model and verify such systems. $\mathsf{LTLf}^{\mathsf{MT}}$ extends LTLf by replacing propositions with first-order formulas over arbitrary theories, *à la* SMT. $\mathsf{LTLf}^{\mathsf{MT}}$ satisfiability is solved with the *symbolic tableau modulo theories* approach and implemented in the state-of-the-art BLACK satisfiability checker, relying on out-of-the-box SMT solvers. This paper recaps such recent developments and discuss open research directions.

Keywords

Linear Temporal Logic, SMT, LTLfMT, Formal Verification, Data-Aware Processes

1. Introduction

Since the seminal work by Pnueli [1] on the one hand, and by Clark and Emerson [2, 3] on the other, the use of specification languages for expressing temporal properties over hardware and software systems has become of prominent importance in computer science. Specifically, traditional model checking techniques [4] focus on the problem of automatically checking whether the hardware or software models of interest meet some given specification represented by some formulae expressed in a suitable formalism. The most common of such languages is surely *Linear Temporal Logic* (LTL) [1]. LTL is interpreted over infinite traces, but its finite-trace counterpart, LTLf [5], has also recently gained traction in AI and *business process modeling* [6], where reasoning over finite traces is more natural since the real execution of a (business) process within a modern organization is assumed to be always finite.

In the context of traditional model checking, the models under investigation are intrinsically *finite-state*: the underlying assumption is that real-world systems can be abstracted by a *finite* representation of their control-flow, disregarding any other source of infiniteness that can arise from the presence of the actual data being manipulated (such as the values of program variables). This abstraction allows one to employ explicit and exhaustive search procedures in the *finite* space of all configurations [7].

OVERLAY 2022: 4th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 28, 2022, Udine, Italy

gianola@inf.unibz.it (A. Gianola); gigante@inf.unibz.it (N. Gigante)

https://gianola.people.unibz.it/ (A. Gianola); https://www.inf.unibz.it/~gigante/ (N. Gigante)

0000-0003-4216-5199 (A. Gianola); 0000-0002-2254-4821 (N. Gigante)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3311/paper3.pdf

This assumption matches with the expressive power of formalisms such as LTL and LTLf, which have in common one limitation: they can be employed to only represent transition systems whose states are finitely many and are characterized by evaluating *propositional* atoms.

However, in real-world scenario the propositional nature of LTL and LTLf can be problematic: for instance, this is the case when modeling *data-aware processes* [8, 9, 10, 11], These are systems whose execution is heavily influenced by the interaction with a persistent data storage such as a relational database: the data storage can modify the behavior of the system, and, vice versa, the system itself can manipulate and update the content of the data storage. In these contexts, one would like to express actions and constraints that depend on the content of the database, which is an inherently first-order, relational model [12].

There is a plethora of similar situations where the data component comes to the picture, turning the systems to the infinite-state case . For instance, one can consider the use case of *data-aware* planning, *i.e.*, planning problems [13] where conditions of actions depend on the content of a relational database. In addition, in the context of business process management applications, some authors have argued for the usefulness of the problem of *structured reinforcement learning* [14], where learning agents operate in an environment where they have access to an explicit structure representing what they know about the world. This explicit structure can be given, for example, by relational data representing the background knowledge of the agents that they can exploit both during and after the learning process, or by some hard constraints that the entire system must satisfy also during learning.

In order to match the expressiveness needed for modeling and reasoning about such scenarios, a new logic called LTLf *Modulo Theories* ($\mathsf{LTLf}^{\mathsf{MT}}$) was recently introduced [15]. This logic extends LTLf by replacing propositional symbols with first-order formulas over an arbitrary theory, in the spirit of *Satisfiability Modulo Theories* (SMT) [16]. While $\mathsf{LTLf}^{\mathsf{MT}}$ is easily seen to be undecidable in general, for decidable theories it is *semi-decidable*, *i.e.*, a positive answer can always be obtained for satisfiable instances, while reasoning might not terminate over unsatisfiable ones. Given the complexity of the problems and scenarios such as those mentioned above, where reasoning is inherently undecidable anyway, being able to solve satisfiable instances is a compromise worth studying.

The satisfiability checking problem for $\mathsf{LTLf}^{\mathsf{MT}}$ is solved by using the *symbolic tableau modulo theory* (STMT) approach [15]. A one-pass and tree-shaped tableau [17, 18], where any *accepted* branch corresponds to a model of the formula, is *symbolically* traversed by means of suitable SMT encodings. This approach allows one to reason on $\mathsf{LTLf}^{\mathsf{MT}}$ by exploiting recent (and future) advancements in SMT solving techniques. The approach has been implemented in the state-of-the-art BLACK tool [19, 20], which builds on top of well-established and highly performing SMT solvers such as Z3 [21] and cvc5 [22], showing promising results [15].

In this paper, we describe $\mathsf{LTLf}^{\mathsf{MT}}$ and discuss the recent results related to this logic, we describe how interesting infinite-state systems can be expressed and verified, and we outline interesting future research directions that this approach naturally opens.

2. LTLf Modulo Theories

LTLf$^{\mathsf{MT}}$ extends LTLf by replacing proposition letters with first-order formulas over arbitrary theories, *à la* Satisfiability Modulo Theories (SMT).

Instead of going into the details of syntax and semantics, let us discuss a few examples:

$$x = 0 \wedge ((\bigcirc x = x + 1)\ \mathsf{U}\ x = 42)$$
$$x = 0 \wedge \mathsf{G}(\widetilde{\bigcirc} x > x \wedge \exists y(x = y + y))$$

Intuitively, the first formula says that the variable x starts at zero, then increments at each time step ($\widetilde{\bigcirc} x = x + 1$), until it reaches the value 42 ($\ldots\ \mathsf{U}\ x = 42$).

The second formula states that the variable x similarly starts at 0, and that in any time point ($\mathsf{G}(\ldots)$), the value of x increases ($\widetilde{\bigcirc} x > x$), and is an even number ($\exists y(x = y + y)$). In other words, x forms a monotonically increasing sequence of even numbers.

The term constructors $\widetilde{\bigcirc} x$ and $\bigcirc x$ are a crucial feature. They both represent, in a different way, the value of x at the *next* time point. The difference lies in how these terms behave at the end of the finite trace. When a first-order formula contains some $\bigcirc x$, the next state is required to exist, while it is not required to exist if it contains only $\widetilde{\bigcirc} x$ terms. This replicates the difference between the *tomorrow* and *weak tomorrow* temporal operators in LTLf [5, 23]. Note that replacing $\widetilde{\bigcirc} x$ with $\bigcirc x$ in the second formula would make it unsatisfiable, since each time point would require the next one, which is impossible in a finite-trace semantics.

The syntax of LTLf$^{\mathsf{MT}}$ is based on arbitrary first-order formulas over a given theory, including quantified formulas, with the restriction that the only temporal operators that can appear inside a quantified formula are *tomorrow* (X) and *weak tomorrow* ($\widetilde{\mathsf{X}}$). First-order variables and next/weak next constructors can be freely combined into arbitrary first-order terms involving function symbols and constants. The semantics of LTLf$^{\mathsf{MT}}$ is defined over *finite* sequences of standard first-order models over the formula's vocabulary. Each function symbol and relation can either be rigid, *i.e.*, fixed over time, or not, *i.e.*, time-evolving.

Satisfiability of LTLf$^{\mathsf{MT}}$ is easily seen to be undecidable in general, but semi-decision procedures can be provided. In particular, LTLf$^{\mathsf{MT}}$ can be semi-decided by a *symbolic tableau modulo theory* (STMT) approach [15]. Given an LTLf$^{\mathsf{MT}}$ formula ϕ, a one-pass and tree-shaped tableau [17, 18] is *symbolically* traversed in a breadth-first fashion by means of an SMT encoding that represents all the accepted branches of the tree of at most length k. If such a formula is satisfiable for some $k > 0$, a model is found. Otherwise, k is incremented. Unsatisfiable formulas may cause this procedure to loop infinitely, as expected for a semi-decidable problem. This procedure has been implemented in the state-of-the-art BLACK[1] satisfiability checker [15].

3. Modeling and Verification

LTLf$^{\mathsf{MT}}$ specifications can be verified against knowledge-base-driven dynamic systems represented by symbolic transition systems whose transitions depend on the first-order states.

[1] https://www.black-sat.org

These systems can be directly expressed as $\mathsf{LTLf}^{\mathsf{MT}}$ formulas (similarly to how standard finite-state transition systems can be expressed in LTL) and the model-checking problem of $\mathsf{LTLf}^{\mathsf{MT}}$ specifications over such systems can be reduced to a $\mathsf{LTLf}^{\mathsf{MT}}$ satisfiability check.

Such models can be employed to approach a diverse variety of real-world data-driven scenarios including *data-aware business processes* [8, 24, 25], database-driven [26, 27] and ontology-driven systems [28, 29, 30], and, in general, any class of first-order-definable infinite-state systems.

Our framework is the first approach capable of representing such an ample set of different systems, while handling general linear-time temporal properties (*i.e.*, not only safety or reachability objectives). Moreover, not only we leverage state-of-the-art SMT solving techniques, but we do so in a robust software tool (BLACK) that is engineered to be ready for real-world usage.

4. Future directions

Many future directions are opened by the introduction of $\mathsf{LTLf}^{\mathsf{MT}}$ and our approach. From a modeling perspective, it is important to classify precisely the kinds of systems discussed in the literature that can be handled by our approach, comparing the efficiency of BLACK with existing tools, such as VERIFAS [27] or MCMT [8, 31, 32]. From an algorithmic perspecive, it is of primary importance to look for particular fragments of $\mathsf{LTLf}^{\mathsf{MT}}$, and/or particular underlying theories, that can result either into a decidable reasoning task, or in more efficient algorithmic techniques. This may include safety/cosafety fragments, particular restrictions to the use of quantifiers (such as the Barneys-Schönfinkel or Rabin classes), restrictions to specific cases of uninterpreted functions/relations (*e.g.*, acyclic relational models), and so on.

While we focused on the satisfiability problem, the *realizability* problem is of foremost interest as well. This is the problem of deciding whether there exists a *strategy* to play a game against an adversary environment in such a way that an $\mathsf{LTLf}^{\mathsf{MT}}$ formula can be guaranteed to be satisfied. This problem, in the case of LTLf specifications, is already very hard (2EXPTIME-complete). In the case of $\mathsf{LTLf}^{\mathsf{MT}}$, should it still be semi-decidable, it would open the doors to a large variety of applications in the context of the *synthesis* of infinite-state systems, rather than of their verification.

Then, applications of $\mathsf{LTLf}^{\mathsf{MT}}$ outside the verification field, and more oriented toward artificial intelligence, are natural developments. In particular, while LTLf can naturally express STRIPS *automated planning problems* [33], $\mathsf{LTLf}^{\mathsf{MT}}$ may be employed to approach a novel class of *data-aware planning* domains. In these models, agents can employ actions whose preconditions depend on the state of a full-fledged relational database, an ontology-based knowledge base, or complex first-order definable data structures (such as lists, trees, graphs, *etc.*), and whose effects actively modify such data containers.

By reducing such data-aware planning problems to $\mathsf{LTLf}^{\mathsf{MT}}$ satisfiability, one may leverage existing reasoning tools such as BLACK to solve them. Moreover, should $\mathsf{LTLf}^{\mathsf{MT}}$ *realizability* turn out to be semi-decidable as well, *fully observable nondeterministic* (FOND) data-aware planning domains could be handled as well. A practically applicable approach to *data-aware FOND planning* is one of the most promising long-term goal of this line of research.

Acknowledgments

This work is partially supported by the Italian Ministry of University and Research under the PRIN program, grant B87G22000450001 (PINPOINT), and by the Free University of Bozen-Bolzano with the SMART-APP, ADAPTERS, and TOTA projects.

References

[1] A. Pnueli, The temporal logic of programs, in: Proc. of FOCS 1977, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.

[2] E. M. Clarke, E. A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: Logics of Programs, Workshop, volume 131 of *Lecture Notes in Computer Science*, Springer, 1981, pp. 52–71. URL: https://doi.org/10.1007/BFb0025774. doi:10.1007/BFb0025774.

[3] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Trans. Program. Lang. Syst. 8 (1986) 244–263. URL: https://doi.org/10.1145/5397.5399. doi:10.1145/5397.5399.

[4] E. M. Clarke, O. Grumberg, D. A. Peled, Model checking, 1st Edition, MIT Press, 2001. URL: http://books.google.de/books?id=Nmc4wEaLXFEC.

[5] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proc. of IJCAI 2013, 2013, pp. 854–860.

[6] G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, M. Montali, Monitoring business metaconstraints based on LTL and LDL for finite traces, in: Proc. of BPM 2014, 2014, pp. 1–17. doi:10.1007/978-3-319-10172-9_1.

[7] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang, Symbolic model checking: 10ˆ20 states and beyond, in: Proc. of (LICS '90, IEEE Computer Society, 1990, pp. 428–439. URL: https://doi.org/10.1109/LICS.1990.113767. doi:10.1109/LICS.1990.113767.

[8] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, SMT-based verification of data-aware processes: a model-theoretic approach, Math. Struct. Comput. Sci. 30 (2020) 271–313. doi:10.1017/S0960129520000067.

[9] Y. Li, A. Deutsch, V. Vianu, VERIFAS: A practical verifier for artifact systems, Proc. VLDB Endow. 11 (2017) 283–296. doi:10.14778/3157794.3157798.

[10] D. Calvanese, G. De Giacomo, M. Montali, Foundations of data-aware process analysis: a database theory perspective, in: Proc. of PODS 2013), ACM, 2013, pp. 1–12.

[11] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Verification of data-aware processes: Challenges and opportunities for automated reasoning, in: Proc. of ARCADE 2019, volume 311, EPTCS, 2019.

[12] B. B. Hariri, D. Calvanese, G. D. Giacomo, A. Deutsch, M. Montali, Verification of relational data-centric dynamic systems with external services, in: Proc. of PODS 2013, ACM, 2013, pp. 163–174. URL: https://doi.org/10.1145/2463664.2465221. doi:10.1145/2463664.2465221.

[13] M. Ghallab, D. S. Nau, P. Traverso, Automated planning - theory and practice, Elsevier, 2004.

[14] A. Gianola, M. Montali, M. Papini, Automated reasoning for reinforcement learning agents

in structured environments, in: Proc. of OVERLAY 2021, volume 2987 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 43–48. URL: http://ceur-ws.org/Vol-2987/paper8.pdf.

[15] L. Geatti, A. Gianola, N. Gigante, Linear temporal logic modulo theories over finite traces, in: Proc. of IJCAI 2022, ijcai.org, 2022, pp. 2641–2647. URL: https://doi.org/10.24963/ijcai.2022/366. doi:$\overline{T}$10.24963/ijcai.2022/366.

[16] C. W. Barrett, C. Tinelli, Satisfiability modulo theories, in: Handbook of Model Checking, Springer, 2018, pp. 305–343. URL: https://doi.org/10.1007/978-3-319-10575-8_11. doi:$\overline{T}$10.1007/978-3-319-10575-8_11.

[17] M. Reynolds, A new rule for LTL tableaux, in: Proc. of GandALF 2016, 2016, pp. 287–301. doi:$\overline{T}$10.4204/EPTCS.226.20.

[18] L. Geatti, N. Gigante, A. Montanari, M. Reynolds, One-pass and tree-shaped tableau systems for TPTL and TPTLb+ past, Information and Computation 278 (2021) 104599.

[19] L. Geatti, N. Gigante, A. Montanari, A SAT-based encoding of the one-pass and tree-shaped tableau system for LTL, in: Proc. of TABLEAUX 2019, 2019, pp. 3–20. URL: https://doi.org/10.1007/978-3-030-29026-9_1. doi:$\overline{T}$10.1007/978-3-030-29026-9_1.

[20] L. Geatti, N. Gigante, A. Montanari, G. Venturato, Past matters: Supporting LTL+Past in the BLACK satisfiability checker, in: Proc. of TIME 2021, 2021.

[21] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: Proc. of TACAS 2008, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340. URL: https://doi.org/10.1007/978-3-540-78800-3_24. doi:$\overline{T}$10.1007/978-3-540-78800-3_24.

[22] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, Y. Zohar, cvc5: A versatile and industrial-strength SMT solver, in: Proc. of TACAS 2022, volume 13243 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 415–442. URL: https://doi.org/10.1007/978-3-030-99524-9_24. doi:$\overline{T}$10.1007/978-3-030-99524-9_24.

[23] G. De Giacomo, R. D. Masellis, M. Montali, Reasoning on LTL on finite traces: Insensitivity to infiniteness, in: Proc. of AAAI 2014, 2014, pp. 1027–1033.

[24] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Formal modeling and SMT-based parameterized verification of data-aware BPMN, in: Proc. of BPM 2019, 2019, pp. 157–175. doi:$\overline{T}$10.1007/978-3-030-26619-6_12.

[25] A. Gianola, SMT-based safety verification of data-aware processes: Foundations and applications, in: Proc. of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2022, volume 3216 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 20–24. URL: http://ceur-ws.org/Vol-3216/paper_133.pdf.

[26] M. Bojanczyk, L. Segoufin, S. Torunczyk, Verification of database-driven systems via amalgamation, in: Proc. of PODS 2013, ACM, 2013, pp. 63–74. doi:$\overline{T}$10.1145/2463664.2465228.

[27] Y. Li, A. Deutsch, V. Vianu, VERIFAS: A practical verifier for artifact systems, Proceedings of the VLDB Endowment 11 (2017) 283–296.

[28] D. Calvanese, A. Gianola, A. Mazzullo, M. Montali, SMT-based safety verification of data-aware processes under ontologies (preliminary results), in: Proc. of DL 2021, volume 2954 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2954/paper-9.pdf.

[29] J. Claßen, M. Liebenberg, G. Lakemeyer, B. Zarrieß, Exploring the boundaries of decidable verification of non-terminating golog programs, in: Proc. of AAAI 2014, AAAI Press, 2014,

pp. 1012–1019. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8625.
[30] B. Zarrieß, J. Claßen, Decidable verification of golog programs over non-local effect actions, in: Proc. of AAAI 2016, AAAI Press, 2016, pp. 1109–1115. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12283.
[31] S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Delta-BPMN: A concrete language and verifier for data-aware BPMN, in: Proc. of BPM 2021, volume 12875 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 179–196. URL: https://doi.org/10.1007/978-3-030-85469-0_13. doi:$\overline{T}$10.1007/978-3-030-85469-0_13.
[32] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Model completeness, uniform interpolants and superposition calculus, J. Autom. Reason. 65 (2021) 941–969. doi:$\overline{T}$10.1007/s10817-021-09596-x.
[33] M. Cialdea Mayer, C. Limongelli, A. Orlandini, V. Poggioni, Linear temporal logic as an executable semantics for planning languages, J. Log. Lang. Inf. 16 (2007) 63–89. doi:$\overline{T}$10.1007/s10849-006-9022-1.