Towards a Comprehensive Formal Language Theory for First-Order Temporal Logic

Nicola Gigante[1]

[1]Free University of Bozen-Bolzano

Abstract

First-Order Linear Temporal Logic (FOLTL) is an extremely expressive formalism that fuses classical *first-order logic* (FO) with the temporal operators of *linear temporal logic* (LTL). This logic has been studied extensively in the knowledge representation community as the formal underpinning of temporal description logics and related formalisms, while its use in reasoning tasks has remained limited due to its undecidability. However, recently the problem of *reactive synthesis* for (fragments of) FOLTL has gained attention from the community, given the potential gain in expressiveness compared with propositional LTL. However, while ω-languages and languages denoted by LTL and $\mathsf{LTL_f}$ (LTL on finite words) are so well understood, a thorough understanding of the *first-order languages* denoted by FOLTL and $\mathsf{FOLTL_f}$ is missing. This missing ingredient would be crucial in a proper development of algorithmic techniques for reactive synthesis of expressive fragments of FOLTL and $\mathsf{FOLTL_f}$. In this communication we highlight the challenges involved in the development of a comprehensive formal language theory for first-order languages, what is known so far, and how we plan to proceed.

Keywords

First-Order Temporal Logic, First-Order Automata, Formal Languages

1. Introduction

First-Order Linear Temporal Logic (FOLTL) combines classical *first-order logic* (FO) with the temporal operators of *linear temporal logic* (LTL), obtaining an extremely expressive formalism. The following is an example FOLTL sentence (over the language and the theory of arithmetics) expressing the property that at each time point the predicate p holds for a superset of the elements where p held before.

$$\mathsf{G}(\forall x(p(x) \rightarrow \mathsf{X}p(x)) \wedge \exists x(\neg p(x) \wedge \mathsf{X}p(x)))$$

Semantically, such a sentence is interpreted over a sequence of *first-order structures* interpreting the quantification domain and the symbols of the language (such as p above) at each time step. Such sequences have been recently called *first-order words* [1] and can be either of infinite or finite length. In the latter case we talk about $\mathsf{FOLTL_f}$.

FOLTL has been studied extensively in the context of first-order modal logics and knowledge representation [2], because it underlies many forms of temporal description logics. However, its use in reasoning tasks has been limited so far, because of its high undecidability: entailment and validity of FOLTL and $\mathsf{FOLTL_f}$ are not even semi-decidable. Only a few and restricted decidable fragments are known, and quite limited in expressiveness.

Nevertheless, recently the community has shown interest in well-behaved fragments of FOLTL and in reasoning about them. One example is $\mathsf{LTL_f}$ *modulo theories* ($\mathsf{LTL_f^{MT}}$) [3], a fragment of $\mathsf{FOLTL_f}$ applying some semantic and syntactic restrictions to gain *semi-decidability* of the satisfiability problem, with an effective semi-decision procedure based on *satisfiability modulo theories* (SMT) [4].

The problem of *reactive synthesis* for $\mathsf{LTL_f^{MT}}$ has been the subject of recent interest. Reactive synthesis is the task of building a correct-by-construction implementation of a reactive system satisfying the given temporal specification. For specifications written in propositional LTL and $\mathsf{LTL_f}$ the problem of deciding

ICTCS 2026: 27th Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy

nicola.gigante@unibz.it (N. Gigante)

https://www.inf.unibz.it/~gigante/ (N. Gigante)

0000-0002-2254-4821 (N. Gigante)

whether such an implementation exists (the problem of *realizability*) is known to be 2EXPTIME-complete [5, 6], while in the case of FOLTL and $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ it is of course undecidable. However, recent developments have shown how to solve the problem for restricted fragments of $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ [7, 8].

These recent development, though, are not easily extensible to larger fragments of FOLTL, not even to the full $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$. We believe a theoretical ingredient is missing that would allow reasoning about FOLTL in more general and rigorous terms, helping the quest for efficient reasoning algorithms (including for reactive synthesis). This ingredient is a thorough understanding of FOLTL and $\mathsf{FOLTL}_\mathsf{f}$ in terms of *formal languages and automata*.

In the propositional world, formal languages and automata theory are the omnipresent theoretical background for any reasoning task in temporal logic. Classically, formal languages and automata have been studied for finite words, where the equivalences between regular expressions and finite-state automata [9] with monadic second-order logic [10] and finite monoids [11] have been proved between the '50s and the '60s of last century. In the same years, LTL and LTL_f have been shown to be equivalent to first-order logic [12], and the latter has been shown to be equivalent to aperiodic monoids [13], which recognise the *star-free* regular languages [14]. Most of these results apply directly to finite or infinite words as well, while some other have been adapted to ω-automata, *i.e.*, automata on infinite words, which have been studied extensively in the subsequent years, especially in the formal verification community.

This huge body of foundational knowledge proved crucial in all the subsequent decades of algorithmic developments in temporal logic and beyond. In contrast, FOLTL has been extensively studied from the perspective of a first-order modal logic, but an understanding of the *first-order languages* (*i.e.*, sets of first-order words) denoted by FOLTL sentences is missing.

In what follows we describes recent developments in the field of reactive synthesis for fragments of FOLTL and $\mathsf{FOLTL}_\mathsf{f}$, and how their improvement would benefit from a comprehensive theory of first-order languages. In Section 2 we provide basic background on reactive synthesis for LTL and LTL_f, in order to introduce first-order reactive synthesis in Section 3. Then, we discuss the recent introduction of *first-order automata* in Section 4, and how they are a starting point in the construction of a general theory of first-order languages. Section 5 concludes with final remarks.

2. Reactive synthesis

Building provably correct and safe systems has always been one of the most challenging tasks in computer science, which has sparked a huge amount of research in the areas of *formal methods* and *software* and *hardware engineering*. One of the most successful paradigms in formal verification is *model checking*, where a specification is checked against an abstract model of the system, looking for bugs in its implementation [15]. An even more challenging task is that of building a *correct-by-construction* implementation of the specification. In the case of the specification of *reactive systems*, *i.e.*, systems that iteratively react to the input from their environment, this task is called *reactive synthesis*.

Reactive synthesis has a long history. The problem was first proposed by Alonzo Church in the '60s of last century [16], and solved by Büchi and Landweber [17] a few years later for specifications written in *monadic second-order logic*. Nowadays, reactive synthesis is usually considered over specifications written in (fragments of) *Linear Temporal Logic*, either on infinite [18] or finite words [19]. The reactive synthesis problem in these cases has been solved by Pnueli and Rosner [5] and De Giacomo and Vardi [6], respectively, with both problems shown to be 2EXPTIME-complete. The task is therefore quite complex, although this complexity has not discouraged research in the area. Indeed, recent decades have seen dramatic developments, with sophisticated techniques and efficient state-of-the-art tools being developed, also fostered by the annual reactive synthesis competition (SYNTCOMP [20]). These developments make reactive synthesis applicable to many domains, from hardware design [21] to the synthesis of declarative business models [22].

A common approach to LTL synthesis is automata-theoretic: a *nondeterministic Büchi automaton* is constructed from the LTL formula in a standard way, which is exponentially larger in the worst

case. Then, the Büchi automaton is determinized into a *deterministic parity automaton*, which is again exponentially larger in the worst case. Then, a parity game can be set up from the parity automaton, which has a winning strategy if and only if the original LTL formula was realizable. From the winning strategy of the parity game one can then extract the actual solution to the reactive synthesis task. Note that determinization is *crucial* because the game can only be used to solve the original problem if formulated on a deterministic arena.

This detour via different kinds of automata and games to solve a 2EXPTIME-complete problem may seem impractical at first, but in fact this is the technique employed and refined by the state-of-the-art synthesis tool Strix [23], which repeatedly won the SYNTCOMP [20] for years. Recent techniques, employed in the SemML tool, which won the last edition [24], combine the classic automata-theoretic techniques with machine learning, obtaining great speedups. Another effective approach proposed in the literature is *bounded synthesis*, implemented *e.g.*, by the BoSy tool [25]. Reactive synthesis of LTL_f specifications is approached with similar automata-theoretic techniques, but with an important difference: the automata involved are not automata on infinite words, but standard finite-state automata on finite words. In particular, the determinization step for nondeterministic finite automata (NFA) to obtain the corresponding deterministic one (DFA) is qualitatively much simpler than the determinization of automata on infinite words, leading to algorithms that, albeit with the same worst-case complexities, scale much better in practice [26]. This led to efficient tools for reactive synthesis of LTL_f specifications, which have been evaluated in recent SYNTCOMP editions as well.

3. Reactive synthesis goes first-order

The expressiveness of LTL and LTL_f is limited by their *propositional* nature, which restricts their usage to the specification of *finite-state* systems. However, many real-world scenarios require to reason about *unbounded*, *complex* and *structured* data, which is impossible to model in LTL and LTL_f.

To model these rich scenarios, the ideal language would be *first-order temporal logic* (FOLTL), which lifts LTL (or LTL_f) from propositional to *first-order logic*. FOLTL has been studied extensively in the knowledge representation community [2] for its great expressiveness, which however does not come for free. All the interesting *reasoning* tasks over FOLTL are *highly undecidable* in general.

Nevertheless, the need for first-order specification languages in reactive synthesis has been recognized in the literature for some time now, and therefore *fragments* of FOLTL have been investigated. The most natural first step is to extend known techniques to handle pieces of unbounded data. This is the case of *temporal stream logic* (TSL) [27, 28], a formalism that extends LTL with the concept of data variables that can be functionally updated during the execution of the system. TSL separates control and data flow, thus recovering decidability and efficient algorithms based on a CEGAR approach (Counterexample-Guided Abstraction Refinement [29]).

Another formalism recently considered for reactive synthesis is a fragment of FOLTL over finite words ($\mathsf{FOLTL}_\mathsf{f}$), that we recently introduced, called LTL_f *modulo theories* ($\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$) [3]. $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ applies syntactic and semantic restrictions to FOLTL: syntactically, it forbids the arbitrary alternation of first-order quantifiers and temporal operators, but allows the use of lookahead and lookback operators that can refer to the value of a variable at the next or previous time step; semantically, it restricts the interpretation of functions and predicates to be rigid, *i.e.*, fixed in time (although arbitrary). When interpreted over a decidable first-order theory, satisfiability of $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ is semi-decidable, and the semi-decision procedure is based on an effective encoding into *satisfiability modulo theories* (SMT), working well in practice on several benchmarks. We also found that, by suitably restricting the theory and/or the syntax, we can recover non-trivial decidable fragments [30].

Reactive synthesis for $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ and LTL^MT was considered by Rodríguez and Sánchez [7], who showed that a Boolean abstraction of an LTL^MT formula can be given to a standard LTL synthesizer to perform reactive synthesis of the fragment *without lookaheads*. Then, they extended their work to lookaheads via a counterexample-guided approach (CEGAR) [31]. Performance is promising, although full potential can be unlocked only with manual intervention, to suggest the system some so-called *reactive tautologies*. A

CEGAR approach is also employed by Azzopardi *et al.* [32] but considering propositional LTL on infinite-state arenas. Then, Winkler [8] provided a direct semi-decision procedure for $\mathsf{LTL}_\mathsf{f}^{\mathsf{MT}}$ synthesis with lookbacks, based on AND-OR graphs and quantifier elimination, although no experimental evaluation was performed and quantifiers are not supported.

4. Beyond the state of the art

The results surveyed above show the interest of the community into reactive synthesis for first-order specifications but also that the field is still in its infancy. The techniques proposed so far, although effective, are quite ad-hoc for their fragments and difficult to lift to more complex scenarios.

A missing key ingredient is a unified theoretical framework on top of which we can ground practical algorithmic development. For LTL and LTL_f, this role has historically been played by *automata theory*. Automata are the theoretical and/or algorithmic tool underlying most successful reactive synthesis techniques for LTL and LTL_f. However, a model of automata capable of capturing *languages* described by FOLTL and its fragments was missing until recently. In fact, a *language-theoretical* treatment of FOLTL is missing as well.

For this reason we recently introduced *first-order automata* [1], a kind of infinite-state, finitely represented automata capable of capturing $\mathsf{FOLTL}_\mathsf{f}$, similarly to how finite-state automata capture LTL_f. In first-order automata, states are first-order structures over a given signature, and the automata are symbolically represented through first-order logic sentences describing their initial states, their transitions, and their final states. We showed that the *first-order languages*, *i.e.*, the set of *first-order words*, satisfying any FOLTL sentence over any first-order theory can be accepted by such automata. By drawing a parallel from standard formal languages on finite alphabets, first-order languages accepted by first-order automata are called *first-order regular*. We showed basic closure properties of first-order regular languages (union, intersection, concatenation, Kleene star) and that *deterministic* first-order automata can be obtained directly from FOLTL sentences employing only *past operators*, which is interesting given the role of determinism in reactive synthesis.

First-order automata are just the first step in a long-term quest for a comprehensive automata and formal language theory for the first-order languages described by FOLTL, which can take the role that automata theory had for propositional reactive synthesis.

Many ingredients are needed to form a comprehensive understanding of these languages. The most immediate open question is whether first-order regular languages are closed under complementation. The common route would pass through determinization of first-order automata, which is still open. Then, expressiveness of $\mathsf{FOLTL}_\mathsf{f}$ w.r.t. the whole class of first-order regular languages needs to be studied. In particular, propositional LTL_f only accepts *star-free languages*, *i.e.*, those denoted by regular expressions employing union, concatenation, complementation but not the Kleene star. Early observations would suggest that $\mathsf{FOLTL}_\mathsf{f}$ as well can only denote a specific subclass of regular languages. For example, similarly to plain LTL_f, denoting all and only the first-order words of *even length* seems to be difficult in $\mathsf{FOLTL}_\mathsf{f}$. Note that $\mathsf{FOLTL}_\mathsf{f}$ sentences over some theory that includes arithmetics *can* count the length of the word and check if its is even. For example:

$$c = 0 \wedge (\exists x.(x = c \wedge \mathsf{X}(c = x + 1))) \,\mathsf{U}\, (\widetilde{\mathsf{X}}\bot \wedge \exists x.c = 2x)$$

The sentence states that a non-rigid constant c has an initial value of zero and increments at each step until the end of the word is reached and the constant's value is even. However, this sentence does *not* characterize all the first-order words of even length (over this signature), but only those of even length where the constant c counts as prescripted. This example seems to suggest that expressiveness limitations similar to those of LTL_f apply to $\mathsf{FOLTL}_\mathsf{f}$ as well, but how to characterise the corresponding subclass of languages, *i.e.*, the first-order counterpart of *star-free languages*, is still open.

Although first-order automata have currently been introduced over finite words, the next natural step is to lift them to infinite words. Extending to FOLTL the existing encoding of $\mathsf{FOLTL}_\mathsf{f}$ into first-order automata is easy, and naturally leads to a form of *first-order Streett automata*. In the propositional

world, Streett automata are just one of a family of ω-automata with different but equivalent acceptance conditions, including Büchi, Rabin, Müller, *etc.* Whether all these acceptance conditions are equivalent in the first-order world is an open question as well.

Another important notion in the world of formal languages which is linked to reactive synthesis is that of *safety* languages. Intuitively, any word that does *not* belong to a safety language, *i.e.*, any *violation* of the corresponding temporal property, has a finite prefix which witness the violation. In the propositional world, reactive synthesis can be computationally easier when starting from LTL formulas of the form $\mathsf{G}\alpha$ where α only employs *past operators* [33], going from 2EXPTIME-complete to EXPTIME-complete. A similar phenomenon may happen for FOLTL, which however requires properly studying the class of safety first-order languages.

5. Conclusions

In this short paper we gave an overview of the state of the art of reactive synthesis for first-order temporal logic which motivates the study of a comprehensive theoretical framework of *first-order formal languages*. We provided an overview of recent results around *first-order automata*, which are a class of infinite-state, finitely representable automata capable of capturing languages denoted by $\mathsf{FOLTL_f}$ sentences, and define the class of *first-order regular languages*. Then, we described the most important ingredients that are still missing in the understanding of these languages.

Given such a theoretical framework, one can envision pushing the state of the art of first-order reactive synthesis to more complex fragments of FOLTL. To do that though, one has to *embrace the enemy*, and concede that *decidable fragments* will be very difficult to find, focusing instead on *semi-decidable* fragments provided with semi-decision procedures that exhibit good behavior *in practice*. This paradigm is not unusual in other fields, for example theorem proving or automated reasoning, which have shown that good heuristics for undecidable problems can prove to be extremely useful in practice.

Algorithmically, supporting fragments more expressive than $\mathsf{LTL_f^{MT}}$ will require radically different techniques. In particular, a limitation is common to $\mathsf{LTL_f^{MT}}$ and all the existing techniques, namely the inability to reason about *non-rigid predicates*, *i.e.*, predicates (*e.g.*, binary relation symbols) whose interpretation change over time. Non-rigid predicates are crucial to model complex scenarios where not only single pieces of data, but their *relationships*, change over time, such as in systems manipulating complex data structures, relational databases, or ontologies.

Supporting non-rigid predicates, however, seems out of reach for all the techniques proposed so far. In a prototypical automata-based forward-search approach to reactive synthesis, the *winning region* is symbolically represented by means of logical formulas or, in the propositional case, binary decision diagrams. Then, *quantifier elimination* is needed at each iteration to recover a ground representation of the winning region. In the case of $\mathsf{LTL_f^{MT}}$, since only *non-rigid constants* (*i.e.*, data variables) change over time, quantifier elimination, when supported by the underlying theory, can be performed with standard techniques. However, when *non-rigid predicates* are involved, a form of *second-order quantifier elimination* (SOQE) is neeeded. SOQE [34], *i.e.*, the elimination of second-order quantifiers from a second-order logic sentence, is a challenging task, which however has been extensively studied in the *knowledge representation* community, because of its many applications in reasoning tasks over ontologies. The application of SOQE is a promising path towards the support of non-rigid predicates in first-order reactive synthesis.

Declaration on Generative AI

The author has not employed any Generative AI tools.

6. Acknowledgements

The author acknowledges the support of the Seal of Excellence Project *Reactive Synthesis goes First-Order* (RESYST, CUP I53C25001920003) funded by the Autonomous Province of Bozen-Bolzano.

References

[1] L. Geatti, A. Gianola, N. Gigante, First-order automata, in: T. Walsh, J. Shah, Z. Kolter (Eds.), Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025, AAAI Press, 2025, pp. 14940–14948. doi:10.1609/AAAI.V39I14.33638.

[2] D. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Fragments of first-order temporal logics, in: Many-Dimensional Modal Logics, volume 148 of *Studies in Logic and the Foundations of Mathematics*, Elsevier, 2003, pp. 465–545. doi:https://doi.org/10.1016/S0049-237X(03)80012-5.

[3] L. Geatti, A. Gianola, N. Gigante, Linear temporal logic modulo theories over finite traces, in: Proceedings of the 31st IJCAI, 2022, pp. 2641–2647. doi:10.24963/ijcai.2022/366.

[4] C. W. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability modulo theories, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 1267–1329. doi:10.3233/FAIA201017.

[5] A. Pnueli, R. Rosner, On the synthesis of an asynchronous reactive module, in: 16th International Colloquium on Automata, Languages and Programming, volume 372 of *Lecture Notes in Computer Science*, Springer, 1989, pp. 652–671. doi:10.1007/BFB0035790.

[6] G. D. Giacomo, M. Y. Vardi, Synthesis for LTL and LDL on finite traces, in: Q. Yang, M. J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 1558–1564. URL: http://ijcai.org/Abstract/15/223.

[7] A. Rodríguez, C. Sánchez, Boolean abstractions for realizability modulo theories, in: Proceedings of the 35th CAV, volume 13966 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 305–328. doi:10.1007/978-3-031-37709-9_15.

[8] S. Winkler, First-order ltlf synthesis with lookback, in: I. Lynce, N. Murano, M. Vallati, S. Villata, F. Chesani, M. Milano, A. Omicini, M. Dastani (Eds.), ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025), Frontiers in Artificial Intelligence and Applications, IOS Press, 2025, pp. 1615–1622. doi:10.3233/FAIA250987.

[9] S. C. Kleene, Representation of events in nerve nets and finite automata, in: C. E. Shannon, J. McCarthy (Eds.), Automata Studies. (AM-34), Princeton University Press, Princeton, 1956, pp. 3–42. doi:10.1515/9781400882618-002.

[10] J. R. Büchi, Weak second-order arithmetic and finite automata, MLQ Math. Log. Q. 6 (1960) 66–92. doi:10.1002/malq.19600060105.

[11] M. Schutzenberger, On an application of semi groups methods to some problems in coding, IRE Trans. Inf. Theory 2 (1956) 47–60. doi:10.1109/tit.1956.1056809.

[12] J. A. W. Kamp, Tense Logic and the Theory of Linear Order, Ph.D. thesis, University of California, Los Angeles, 1968.

[13] R. McNaughton, S. A. Papert, Counter-Free Automata (M.I.T. research monograph no. 65), The MIT Press, 1971.

[14] M. Schützenberger, On finite monoids having only trivial subgroups, Inf. Contr. 8 (1965) 190–194. doi:10.1016/S0019-9958(65)90108-7.

[15] E. M. Clarke, O. Grumberg, D. Kroening, D. A. Peled, H. Veith, Model checking, 2nd Edition, MIT Press, 2018.

[16] A. Church, Application of recursive arithmetic to the problem of circuit synthesis, Journal of Symbolic Logic 28 (1963) 289–290. doi:10.2307/2271310.

[17] J. R. Büchi, L. H. Landweber, Definability in the monadic second-order theory of successor, J. Symb. Log. 34 (1969) 166–170. doi:10.2307/2271090.

[18] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1977, pp. 46–57. URL: https://doi.org/10.1109/SFCS.1977.32. doi:10.1109/SFCS.1977.32.

[19] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of 23rd IJCAI, 2013, pp. 854–860. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

[20] S. Jacobs, G. A. Pérez, R. Abraham, V. Bruyère, M. Cadilhac, M. Colange, C. Delfosse, T. van Dijk, A. Duret-Lutz, P. Faymonville, B. Finkbeiner, A. Khalimov, F. Klein, M. Luttenberger, K. J. Meyer, T. Michaud, A. Pommellet, F. Renkin, P. Schlehuber-Caissier, M. Sakr, S. Sickert, G. Staquet, C. Tamines, L. Tentrup, A. Walker, The reactive synthesis competition (SYNTCOMP): 2018-2021, Int. J. Softw. Tools Technol. Transf. 26 (2024) 551–567. doi:10.1007/S10009-024-00754-1.

[21] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, Y. Sa'ar, Synthesis of reactive(1) designs, J. Comput. Syst. Sci. 78 (2012) 911–938. doi:10.1016/J.JCSS.2011.08.007.

[22] L. Geatti, M. Montali, A. Rivkin, Foundations of reactive synthesis for declarative process specifications, in: M. J. Wooldridge, J. G. Dy, S. Natarajan (Eds.), Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, AAAI Press, 2024, pp. 17416–17425. doi:10.1609/AAAI.V38I16.29690.

[23] P. J. Meyer, S. Sickert, M. Luttenberger, Strix: Explicit reactive synthesis strikes back!, in: H. Chockler, G. Weissenbacher (Eds.), Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I, Lecture Notes in Computer Science, Springer, 2018, pp. 578–586. doi:10.1007/978-3-319-96145-3_31.

[24] J. Kretínský, T. Meggendorfer, M. Prokop, A. Zarkhah, Semml: Enhancing automata-theoretic LTL synthesis with machine learning, in: A. Gurfinkel, M. Heule (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 31st International Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I, Lecture Notes in Computer Science, Springer, 2025, pp. 233–253. doi:10.1007/978-3-031-90643-5_12.

[25] P. Faymonville, B. Finkbeiner, L. Tentrup, Bosy: An experimentation framework for bounded synthesis, in: R. Majumdar, V. Kuncak (Eds.), Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II, Lecture Notes in Computer Science, Springer, 2017, pp. 325–332. doi:10.1007/978-3-319-63390-9_17.

[26] S. Xiao, Y. Li, S. Zhu, J. Sun, J. Li, G. Pu, M. Y. Vardi, An on-the-fly synthesis framework for LTL over finite traces, ACM Trans. Softw. Eng. Methodol. 35 (2026) 115:1–115:33. doi:10.1145/3749101.

[27] B. Finkbeiner, F. Klein, R. Piskac, M. Santolucito, Temporal stream logic: Synthesis beyond the bools, in: I. Dillig, S. Tasiran (Eds.), Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I, Lecture Notes in Computer Science, Springer, 2019, pp. 609–629. doi:10.1007/978-3-030-25540-4_35.

[28] B. Finkbeiner, P. Heim, N. Passing, Temporal stream logic modulo theories, in: P. Bouyer, L. Schröder (Eds.), Foundations of Software Science and Computation Structures - 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Lecture Notes in Computer Science, Springer, 2022, pp. 325–346. doi:10.1007/978-3-030-99253-8_17.

[29] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement, in: Proceedings of 12th CAV, volume 1855 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 154–169. doi:10.1007/10722167_15.

[30] L. Geatti, A. Gianola, N. Gigante, S. Winkler, Decidable fragments of ltl_f modulo theories, in: Proceedings of the 26th ECAI, volume 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 811–818. doi:10.3233/FAIA230348.

[31] A. Rodríguez, F. Gorostiaga, C. Sánchez, Counter example guided reactive synthesis for LTL modulo theories*, in: R. Piskac, Z. Rakamaric (Eds.), Computer Aided Verification - 37th International Conference, CAV 2025, Zagreb, Croatia, July 23-25, 2025, Proceedings, Part IV, Lecture Notes in Computer Science, Springer, 2025, pp. 224–248. doi:10.1007/978-3-031-98685-7_11.

[32] S. Azzopardi, L. D. Stefano, N. Piterman, G. Schneider, Full LTL synthesis over infinite-state arenas, in: R. Piskac, Z. Rakamaric (Eds.), Computer Aided Verification - 37th International Conference, CAV 2025, Zagreb, Croatia, July 23-25, 2025, Proceedings, Part IV, Lecture Notes in Computer Science, Springer, 2025, pp. 274–297. doi:10.1007/978-3-031-98685-7_13.

[33] A. Artale, L. Geatti, N. Gigante, A. Mazzullo, A. Montanari, Complexity of safety and cosafety fragments of linear temporal logic, in: Proceedings of AAAI 2023, AAAI Press, 2023, pp. 6236–6244. doi:10.1609/AAAI.V37I5.25768.

[34] D. M. Gabbay, R. A. Schmidt, A. Szalas, Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications, volume 12 of *Studies in logic : Mathematical logic and foundations*, College Publications, 2008.