Linear Temporal Logic – From Infinite to Finite Horizon

Lucas M. Tabajara and Moshe Y. Vardi(✉)

Rice University, Houston, USA
vardi@rice.edu

Abstract. Linear Temporal Logic (LTL), proposed by Pnueli in 1977 for reasoning about ongoing programs, was defined over infinite traces. The motivation for this was the desire to model arbitrarily long computations. While this approach has been highly successful in the context of model checking, it has been less successful in the context of reactive synthesis, due to the challenging algorithmics of infinite-horizon temporal synthesis. In this paper we show that focusing on finite-horizon temporal synthesis offers enough algorithmic advantages to compensate for the loss in expressiveness. In fact, finite-horizon reasonings is useful even in the context of infinite-horizon applications.

1 Reactive Systems and Reactive Synthesis

Reactive systems are widespread in modern society, from our personal computers to traffic control systems and factory robots, and we can expect them to become even more ubiquitous with the recent advent of new technologies such as autonomous vehicles and Internet of Things. A reactive system is any kind of computer system that operates in a continuous loop interacting with an external environment. This environment can be the physical world, another component of a larger system, or other systems connected in a network [25].

Because reactive systems interact with other systems and the real world, it is especially important to guarantee that such systems operate safely and correctly, since errors in their operation can have far reaching and often serious consequences. But designing such systems correctly can be especially challenging, since they can run for an unbounded amount of time, and their internal state at any given moment depends on the entire history of inputs received since they started operation. Therefore, the designer has to make sure that they respond correctly to a potentially infinite set of possible environment behaviors [25].

This challenge motivates the problem of *reactive synthesis* [40], which proposes an alternative to the manual design of reactive systems. Instead, reactive synthesis aims to automatically and algorithmically generate a reactive system from a specification of its desired behavior. This specification is usually given as

Work supported in part by NSF grants IIS-1527668, CCF-1704883,IIS-1830549, DoD MURI grant N00014-20-1-2787, and an award from the Maryland Procurement Office.

Z. Hou and V. Ganesh (Eds.): ATVA 2021, LNCS 12971, pp. 3–12, 2021.
https://doi.org/10.1007/978-3-030-88885-5_1

a formula in some type of temporal logic expressing the set of acceptable execution traces of the system. A reactive system is said to *realize* this specification if every execution trace produced by the system satisfies the formula, regardless of the inputs received from the environment. A reactive synthesis algorithm should be able to determine if the specification is realizable and, if so, synthesize a system realizing it.

Possibly the most common specification language for reactive synthesis is Linear Temporal Logic (LTL) [39], an extension of propositional logic with temporal operators such as "next", "until", "eventually" and "globally". The classic approach for reactive synthesis from an LTL specification is based on reducing the problem to a game played over a deterministic ω-automaton [42], a class of automata that accept languages over infinite words. This approach proceeds as follows:

1. Convert the LTL formula into some type of deterministic ω-automaton, such as a deterministic Rabin [42,43] or parity [20] automaton, that accepts exactly the language of traces that satisfy the formula.
2. Use the automaton as the arena for a two-player game between the system and the environment, where the system wins if it satisfies the acceptance condition of the automaton [40].
3. Solve the game to find out which player has a winning strategy (such games are always determined) [24]. If the system wins, the specification is realizable and the winning strategy can be used as a model for the reactive system.

Over the years, reactive synthesis has been extensively studied in the field of formal methods. Yet, not much of the progress in the area has translated into making reactive synthesis significantly more practical for real-world applications [31]. Techniques like bounded synthesis [19,21–23], symbolic algorithms [7,19] and on-the-fly game construction [35] have made implementation of synthesis algorithms more feasible, but tools for reactive synthesis still have limited scalability, which has largely prevented this problem from gaining traction for practical applications. Furthermore, many generalizations of reactive synthesis that would be of interest in real-world scenarios, such as quantitative synthesis [1,2] and synthesis with incomplete information [32], have not been able to be fully explored in practice so far, since they layer additional complexity on top of a problem that is already challenging to solve efficiently.

At first glance, it is easy to attribute this lack of practical impact to the worst-case complexity of reactive synthesis: the problem is 2EXPTIME-complete, meaning that deciding whether a specification is realizable may take doubly-exponential time in the size of the formula [41]. But this complexity analysis can be deceptive. First, it considers only the worst case, which rarely occurs in practice. In fact, many useful classes of specification can be synthesized in exponential or even polynomial time [4,6,12]. Second, the doubly-exponential upper bound speaks more of the succinctness of LTL as a specification language than anything else: some classes of LTL formulas specify properties that can only be realized by a system of doubly-exponential size [41].

Instead of the worst-case complexity, it might make more sense to attribute the poor practical performance of reactive-synthesis algorithms to the lack of efficient algorithms for ω-automata. Other applications of such automata in formal methods, like LTL model checking, tend to use nondeterministic Büchi automata (NBA) [48]. The classic approach to reactive synthesis, however, requires a *deterministic* automaton [40], which leads to a number of complications. Unlike NBAs, deterministic Büchi automata are not expressive enough to represent all LTL formulas [30], forcing determinization to produce an automaton with a more complex acceptance condition, such as a *parity automaton*. The classic algorithm for performing this procedure is Safra's algorithm [38,42,43], which is notoriously complex and difficult to understand, let alone implement efficiently [3,47]. Furthermore, it is still an open problem whether games over parity automata can be solved in polynomial time [9], an issue that is compounded by the complexity of the state space generated by Safra's construction.

These observations suggest that in order for reactive synthesis to be efficiently implementable in practice, it is necessary to overcome the algorithmic barriers imposed by ω-automata. One of the most successful attempts to do this has been Generalized Reactivity(1) (GR(1)) synthesis [6], which has become maybe the only variant of reactive synthesis that has achieved widespread use in application domains, particularly robotics [29,36]. Despite GR(1) being a more limited specification format, GR(1) synthesis has a number of advantages over synthesis from LTL specifications [6]:

- The state space of the game is directly encoded in the GR(1) specification, thus entirely avoiding having to use automaton construction and determinization.
- The winning condition of the game is a GR(1) condition, which is more general than a Büchi condition, but simpler than a parity condition. Unlike parity games, games with a GR(1) condition can be solved in polynomial (cubic) time.
- The game can be naturally represented symbolically, as states correspond to assignments to Boolean variables and the transition relation can be represented as a Boolean formula. This enables the use of efficient symbolic algorithms. In contrast, the games produced by Safra's construction have very complex state spaces which are not amenable to a symbolic encoding.

The success story of GR(1) demonstrates how, by trying to do less, we can accomplish more: by imposing limits on what types of problems can be specified and how, it becomes possible to attain a synthesis procedure that has hopes of being useful in practice. GR(1) achieves this by entirely avoiding the use of automata, but a more recently-proposed variant of reactive synthesis brings to light an alternative option: replacing ω-automata by the simpler and more tractable automata over finite words.

2 LTL Synthesis over Finite Traces

Classically, LTL is interpreted over infinite traces, which is consistent with the idea that a reactive system might operate continuously for an indeterminate amount of time [25]. Many applications, however, use LTL to specify finite-horizon behaviors, especially in areas such as robotics and planning in AI, where systems more often than not have a finite-horizon task to complete [11,26,37]. This has led to the formalization of LTL with finite-trace semantics, or LTL_f [16].

Reactive synthesis from LTL_f specifications has found promising applications in planning and robotics, where it is closely related to fully-observable nondeterministic (FOND) planning [10,15]. In this context, LTL_f synthesis can be used to synthesize a policy for an autonomous agent to complete a task within an unpredictable environment. An example is when a robot needs to complete a task in the presence of humans, who can both aid and interfere in the completion of the task [26,49]. Synthesis can be used to generate a policy that considers all possible behaviors of the humans and other components of the environment (within a set of assumptions) and chooses how to respond to each in order to fulfill the task. Using synthesis thus avoids the need for re-planning in the case of an uncooperative environment [33].

What makes LTL_f promising in the context of reactive synthesis is that it opens up the possibility of algorithms based on automata over *finite*, rather than infinite, words. LTL_f has the same expressive power as first-order logic (FOL) over finite sequences [16]. Both are strictly less expressive than monadic second-order logic (MSO) over finite sequences, which is equivalent to finite automata [8]. This means that every LTL_f formula can be converted into a deterministic finite automaton (DFA) that accepts exactly those finite traces that satisfy the formula.

As a consequence, when the specification can be expressed as an LTL_f formula, reactive synthesis can be solved using an algorithm based on DFAs instead than ω-automata: the LTL_f formula is converted into an equivalent DFA, and the system is synthesized by solving a reachability game over this DFA [17]. Although LTL_f synthesis has the same 2EXPTIME-complete complexity as LTL synthesis, this DFA-based algorithm has a number of advantages in relation to the classic algorithms for LTL synthesis [53]:

1. Determinization of automata over finite words can use the classic subset construction algorithm, which despite still being exponential is much simpler and more efficient than Safra's construction, as well as being very amenable to symbolic representation.
2. DFA minimization is much more viable than for ω-automata. While minimization of ω-automata is NP-complete [44], DFAs have a minimal canonical form, which can be computed efficiently in time $O(n \log n)$ [28].
3. Reachability games are much simpler than parity games, being solvable in linear time [34].

We have used this theoretical algorithm as the basis for Syft, the first framework for performing LTL_f synthesis in practice [53], which takes advantage of the benefits of DFAs outlined above. Syft works in the following way:

1. The LTL_f specification is converted into an equivalent formula in FOL over finite traces.
2. The FOL formula is given as input to the tool Mona [27], which constructs the minimal DFA for the language of the formula.
3. The DFA is converted to a compact symbolic representation, using Binary Decision Diagrams (BDDs) to represent the state sets and the transition function.
4. A reachability game is solved over this DFA using a symbolic fixpoint algorithm that constructs a BDD representing the set of winning states. If the game is winning for the system, a winning strategy is constructed using BDDs as well.

Our empirical results showed that, despite LTL_f synthesis having the same 2EXPTIME complexity as LTL synthesis, Syft performed far better in practice than converting the LTL_f specification to an equivalent LTL formula and giving it as input to existing tools for LTL synthesis [53]. This difference in performance can be attributed to the benefits of DFAs previously mentioned:

- The reachability game played on the DFA can be solved in linear time, and the symbolic implementation further improves the performance.
- Thanks to the ease of DFA minimization, Mona is able to output a fully-minimized DFA, decreasing the state space of the reachability game and making it easier to solve.
- Mona furthermore constructs the DFA in stages, minimizing intermediate DFAs. This leads to better performance in terms of both time and memory for the DFA construction [51].

Despite the differences between the two approaches, LTL_f synthesis is able to benefit from the same strategy as GR(1) synthesis: by limiting the scope of the problem (in this case, to finite-horizon tasks) it becomes possible to achieve success where classic reactive synthesis failed. The advantages are similar to those for GR(1): avoiding the expensive determinization of ω-automata, reducing the problem to a game that can be solved in polynomial time, and producing a simpler and more compact state space that is amenable to a symbolic representation. LTL_f synthesis was able to achieve this by replacing the classic algorithms based on ω-automata with DFA-based methods, and the initial experiments using Syft have demonstrated the potential of this approach.

It is only natural to now ask what other doors DFA algorithms have opened for reactive synthesis. For instance, can DFAs be used also for synthesis over infinite traces? Can we design better algorithms for constructing and manipulating DFAs in order to improve synthesis performance? And now that DFAs have allowed us to reach an algorithm with more practical potential, can we generalize it to extensions of reactive synthesis like synthesis with incomplete information, which were previously infeasible to explore in practice? These are some of the questions that our work seeks to answer.

3 Synthesis Using Finite-Word Automata

Over the last few years we have focused on several research directions on the topic of DFA-based approaches for reactive synthesis.

One such line of research is exploring how DFA algorithms can be extended beyond synthesis over finite traces into synthesis of infinite traces, by identifying classes of specifications involving infinite-trace semantics for which the synthesis problem can similarly be reduced to a game over a DFA. In this way, the algorithmic benefits of DFAs can be exploited also for these types of specifications. One such setting is synthesis of Safety LTL [52], a fragment of LTL that can only express safety properties, meaning properties where every violation occurs in a finite time. As a consequence, the synthesis problem for this fragment can be reduced to a safety game, the dual of a reachability game, which likewise can be solved in linear time. Furthermore, the arena for this game can be constructed as a DFA for the language of finite prefixes that violate the specification, allowing us to take advantage of Mona and the efficient algorithms for DFA construction. Another example is LTL_f augmented with infinite-trace assumptions, including LTL and GR(1) assumptions [13,14,50]. In this line of work, the task the system has to complete is finite, but its satisfaction might depend on an assumption of infinite behavior on the part of the environment. This means that the system might need to wait an unbounded amount of time for the right conditions to complete its task. Similarly to Safety LTL, this class of specifications can lead to games that are simpler than parity games, for example, GR(1) games, and where the arena can also be constructed as a DFA. For both Safety LTL and infinitary-assumption LTL_f, the DFA-based algorithms outperform the use of tools for LTL synthesis.

Another direction focuses on attempting to improve DFA construction in a way that can lead to better performance of synthesis algorithms, based on the fact that early experiments indicated that DFA construction was the bottleneck of the Syft pipeline [53]. In [46] we presented a solution that avoids the cost of constructing the full DFA explicitly by instead representing the reachability game by the implicit product of smaller DFAs. The experimental results showed, however, that although the construction of this partitioned game is more efficient, it does not compensate for the overhead incurred for solving the game over this representation. A deeper analysis identified the root cause of the issue to be the fact that, although the partitioned game is a more compact representation, it prevents taking full advantage of DFA minimization, leading to an enlarged implicit state space that makes the reachability game harder to solve. The insights obtained from the results and analysis in that work later allowed the design an improved algorithm that achieves a balance between partitioning and minimization [5].

Finally, in [45] we investigated how the properties of automata over finite words affect the performance in practice of LTL_f synthesis under partial observability, a generalization of standard LTL_f synthesis where the system must satisfy the specification even in the presence of unobservable inputs [18]. Our work presented the first practical implementation of synthesis under partial observabil-

ity, making use of Mona and symbolic techniques to integrate two previously-proposed algorithms for partial observability [18] into the Syft framework. In addition, a third algorithm was introduced that emerges naturally from the use of Mona for DFA construction. The empirical evaluation showed that the practical performance of the algorithms differs significantly from what the theoretical complexity analysis predicts, due to the absence in practice of the worst-case exponential gap between deterministic and nondeterministic finite automata. These results demonstrated that, especially when dealing with finite automata, worst-case complexity is not necessarily a good predictor of practical performance, highlighting the importance of complementing theoretical analysis with an experimental evaluation.

Each of these three research directions contributes to exploring the full potential of approaches based on automata over finite words for reactive synthesis. The results improve on the state of the art and demonstrate the benefits of DFA-based algorithms, such as the value of state-space minimization for synthesis performance. The insights obtained from these works will hopefully be useful as a guide for future research on DFA-based synthesis.

References

1. Almagor, S., Boker, U., Kupferman, O.: Formally reasoning about quality. J. ACM **63**(3), 24:1–24:56 (2016). https://doi.org/10.1145/2875421
2. Almagor, S., Kupferman, O.: High-quality synthesis against stochastic environments. In: Talbot, J., Regnier, L. (eds.) CSL. LIPIcs, vol. 62, pp. 28:1–28:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016)
3. Althoff, C.S., Thomas, W., Wallmeier, N.: Observations on determinization of Büchi automata. In: Farré, J., Litovsky, I., Schmitz, S. (eds.) CIAA 2005. LNCS, vol. 3845, pp. 262–272. Springer, Heidelberg (2006). https://doi.org/10.1007/11605157_22
4. Alur, R., Torre, S.L.: Deterministic generators and games for LTL fragments. In: IEEE, pp. 291–300. IEEE Computer Society (2001). https://doi.org/10.1109/LICS.2001.932505
5. Bansal, S., Li, Y., Tabajara, L.M., Vardi, M.Y.: Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications. In: AAAI, pp. 9766–9774 (2020)
6. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)
7. Bohy, A., Bruyère, V., Filiot, E., Jin, N., Raskin, J.-F.: Acacia+, a tool for LTL synthesis. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 652–657. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_45
8. Büchi, J.R.: Weak second-order arithmetic and finite automata. In: Mac, L.S., Siefkes, D. (eds.) The Collected Works of J. Richard Büchi, pp. 398–424. Springer, New York (1990). https://doi.org/10.1007/978-1-4613-8928-6_22
9. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: STOC, pp. 252–263 (2017)
10. Camacho, A., Baier, J.A., Muise, C.J., McIlraith, S.A.: Finite LTL synthesis as planning. In: ICAPS, pp. 29–38 (2018)

11. Camacho, A., Triantafillou, E., Muise, C., Baier, J.A., McIlraith, S.: Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In: AAAI, pp. 3716–3724 (2017)
12. Cheng, C.-H., Hamza, Y., Ruess, H.: Structural synthesis for GXW specifications. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 95–117. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_6
13. De Giacomo, G., Di Stasio, A., Tabajara, L.M., Vardi, M., Zhu, S.: Finite-trace and generalized-reactivity specifications in temporal synthesis. In: IJCAI (2021)
14. De Giacomo, G., Di Stasio, A., Vardi, M.Y., Zhu, S.: Two-stage technique for LTL_f synthesis under LTL assumptions. In: KR (2020)
15. De Giacomo, G., Rubin, S.: Automata-theoretic foundations of FOND planning for LTL_f/LDL_f Goals. In: IJCAI, pp. 4729–4735 (2018)
16. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI, pp. 854–860 (2013)
17. De Giacomo, G., Vardi, M.Y.: Synthesis for LTL and LDL on finite traces. In: IJCAI, pp. 1558–1564 (2015)
18. De Giacomo, G., Vardi, M.Y.: LTL_f and LDL_f synthesis under partial observability. In: IJCAI, pp. 1044–1050 (2016)
19. Ehlers, R.: Unbeast: symbolic bounded synthesis. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 272–275. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19835-9_25
20. Emerson, E., Jutla, C.: On simultaneously determinizing and complementing ω-automata. In: Proceedings of 4th IEEE Symposium on Logic in Computer Science, pp. 333–342 (1989)
21. Faymonville, P., Finkbeiner, B., Rabe, M.N., Tentrup, L.: Encodings of bounded synthesis. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10205, pp. 354–370. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54577-5_20
22. Faymonville, P., Finkbeiner, B., Tentrup, L.: BoSy: an experimentation framework for bounded synthesis. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 325–332. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_17
23. Finkbeiner, B., Schewe, S.: Bounded synthesis. Int. J. Softw. Tools Technol. Transf. **15**(5–6), 519–539 (2013). https://doi.org/10.1007/s10009-012-0228-z
24. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]. Lecture Notes in Computer Science, vol. 2500. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36387-4
25. Harel, D., Pnueli, A.: On the development of reactive systems. In: Apt, K. (ed.) Logics and Models of Concurrent Systems, NATO Advanced Summer Institutes, vol. 13, pp. 477–498. Springer, Heidelberg (1985). https://doi.org/10.1007/978-3-642-82453-1_17
26. He, K., Wells, A.M., Kavraki, L.E., Vardi, M.Y.: Efficient symbolic reactive synthesis for finite-horizon tasks. In: ICRA, pp. 8993–8999. IEEE (2019)
27. Henriksen, J.G., et al.: MONA: monadic second-order logic in practice. In: TACAS, pp. 89–110 (1995)
28. Hopcroft, J.: An $n \log n$ algorithm for minimizing states in a finite automaton. In: Theory of machines and computations, pp. 189–196. Elsevier (1971)
29. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal-logic-based reactive mission and motion planning. IEEE Trans. Rob. **25**(6), 1370–1381 (2009)

30. Krishnan, S.C., Puri, A., Brayton, R.K.: Deterministic ω automata vis-a-vis deterministic Buchi automata. In: Du, D.-Z., Zhang, X.-S. (eds.) ISAAC 1994. LNCS, vol. 834, pp. 378–386. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58325-4_202
31. Kupferman, O.: Recent challenges and ideas in temporal synthesis. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 88–98. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27660-6_8
32. Kupferman, O., Vardi, M.: Synthesis with incomplete informatio. In: ICTL, pp. 1044–1050 (1997)
33. Lahijanian, M., Maly, M.R., Fried, D., Kavraki, L.E., Kress-Gazit, H., Vardi, M.Y.: Iterative temporal planning in uncertain environments with partial satisfaction guarantees. IEEE Trans. Robot. **32**(3), 583–599 (2016)
34. Mazala, R.: Infinite games. In: Grädel, E., Thomas, W., Wilke, T. (eds.) Automata Logics, and Infinite Games. LNCS, vol. 2500, pp. 23–38. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36387-4_2
35. Meyer, P.J., Sickert, S., Luttenberger, M.: Strix: explicit reactive synthesis strikes back! In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 578–586. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_31
36. Moarref, S., Kress-Gazit, H.: Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. Auton. Robot. **44**(3–4), 585–600 (2020). https://doi.org/10.1007/s10514-019-09861-4
37. Pešić, M., Bošnački, D., van der Aalst, W.M.P.: Enacting declarative languages using LTL: avoiding errors and improving performance. In: van de Pol, J., Weber, M. (eds.) SPIN 2010. LNCS, vol. 6349, pp. 146–161. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16164-3_11
38. Piterman, N.: From nondeterministic Büchi and streett automata to deterministic parity automata. Log. Methods Comput. Sci. **3**(3) (2007)
39. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, pp. 46–57 (1977)
40. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190 (1989)
41. Rosner, R.: Modular synthesis of reactive systems. Ph.D. thesis, The Weizmann Institute of Science (1991)
42. Safra, S.: On the complexity of ω-automata. In: FOCS, pp. 319–327 (1988)
43. Safra, S.: Exponential determinization for ω-automata with a strong fairness acceptance condition. SIAM J. Comput. **36**(3), 803–814 (2006)
44. Schewe, S.: Beyond hyper-minimisation–minimising DBAs and DPAs is NP-complete. In: Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science. LIPIcs, vol. 8, pp. 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010)
45. Tabajara, L.M., Vardi, M.Y.: LTL_f synthesis under partial observability: from theory to practice. In: Raskin, J., Bresolin, D. (eds.) GandALF. EPTCS, vol. 326, pp. 1–17 (2020). https://doi.org/10.4204/EPTCS.326.1
46. Tabajara, L.M., Vardi, M.Y.: Partitioning techniques in LTL_f synthesis. In: IJCAI, pp. 5599–5606. AAAI Press (2019)
47. TaŞiran, S., Hojati, R., Brayton, R.K.: Language containment of non-deterministic ω-automata. In: Camurati, P.E., Eveking, H. (eds.) CHARME 1995. LNCS, vol. 987, pp. 261–277. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60385-9_16

48. Vardi, M.Y.: Automata-theoretic model checking revisited. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 137–150. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69738-1_10
49. Wells, A.M., Lahijanian, M., Kavraki, L.E., Vardi, M.Y.: LTL_f synthesis on probabilistic systems. In: Raskin, J., Bresolin, D. (eds.) GandALF. EPTCS, vol. 326, pp. 166–181 (2020). https://doi.org/10.4204/EPTCS.326.11
50. Zhu, S., De Giacomo, G., Pu, G., Vardi, M.Y.: LTL_f synthesis with fairness and stability assumptions. In: AAAI, pp. 3088–3095 (2020)
51. Zhu, S., Pu, G., Vardi, M.Y.: First-order vs. second-order encodings for LTL_f-to-automata translation. In: TAMC, pp. 684–705 (2019)
52. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: A symbolic approach to safety LTL synthesis. In: HVC 2017. LNCS, vol. 10629, pp. 147–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70389-3_10
53. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: Symbolic LTL_f synthesis. In: IJCAI, pp. 1362–1369 (2017)