Reactive Synthesis for DECLARE Process Models

Luca Geatti[1] **Marco Montali,** [2] **Andrey Rivkin,** [2]

[1] Università degli Studi di Udine, Udine, Italy
[2] Free University of Bozen-Bolzano, Bolzano, Italy
luca.geatti@uniud.it, {montali,rivkin}@inf.unibz.it

Introduction

Since its introduction by Pnueli (Pnueli 1977), Linear Temporal Logic (LTL) has been extensively employed in a variety of verification and synthesis tasks. While verification aims at checking the correctness of an LTL specification over a given dynamic system, synthesis uses an LTL specification to derive a corresponding correct-by-construction program (in the shape, *e.g.*, of a Mealy or Moore machine) that realizes the specification. Extensive research has been conducted on different synthesis settings, considering in particular closed and *open* (also called *reactive*) systems, starting from the seminal contributions by Harel, Pnueli and Rosner (Harel and Pnueli 1984; Pnueli and Rosner 1989a). In the reactive setting, the system (Controller) interacts with Environment, which, in turn, can affect the behavior of Controller. Like that, reactive synthesis can be represented as a two-player game between Controller, whose aim is to satisfy the formula, and Environment, who tries to violate it. The objective of the synthesis task is then to synthesize a program for Controller indicating which actions the Controller should take to guarantee the satisfaction of the LTL specification of interest, no matter what are the actions taken by Environment. This problem was originally studied in (Church 1962) and the first solution for it was proposed in (Buchi and Landweber 1990). Notably, for LTL specifications the synthesis problem was shown to be 2EXPTIME-complete (Pnueli and Rosner 1989b; Rosner 1992). The high theoretical complexity and practical infeasibility of the original approach inspired the scientific community to search for fragments of LTL that have many practical applications and for which the synthesis problem is computationally more feasible. Generalized Reactivity(1) formulas (Bloem et al. 2012) make a good example of these fragments, as the synthesis problem can be solved in cubic time for such formulas.

Recently, a lot of attention has been put on LTLf (De Giacomo and Vardi 2013) – LTL interpreted over *finite* traces. This logic allows to represent the system dynamics consisting of traces of unbounded, yet finite length (Giacomo, Masellis, and Montali 2014), and has been extensively studied within AI, formal methods, and business process management (De Giacomo and Vardi 2015; Pesic and van der Aalst 2006; De Giacomo, De Masellis, and Montali 2014; Fionda and Greco 2018; Artale, Mazzullo, and Ozaki 2019; De Giacomo et al. 2019; Maggi, Montali, and Peñaloza 2020; De Giacomo et al. 2022). A lot of progress has been made in studying LTLf-based synthesis (De Giacomo and Vardi 2015; Zhu et al. 2017; Camacho et al. 2018; Zhu, Pu, and Vardi 2019; Bansal et al. 2020; Zhu et al. 2020; Giacomo et al. 2020; Xiao et al. 2021; Giacomo et al. 2022), where the synthesised program always terminates (differently from the infinite-trace case). Surprisingly, the theoretical complexity of the synthesis problem for LTLf remains 2EXPTIME-complete (De Giacomo and Vardi 2015). This can be witnessed by the following three steps taken by the classical LTLf synthesis algorithm: *(1)* encode the formula into a nondeterministic automaton over finite words (NFA); *(2)* create its deterministic version (DFA); *(3)* solve a *reachability game*, in which the objective of Controller is to force the game to reach a final state of the DFA. Each one of the first two steps requires, in the worst case, an exponential amount of time in the size of its input.

In this work, we start from the conjecture that it is possible to find a fragment of LTLf that is widely used in practice and for which the synthesis problem enjoys better computational complexity. To this end, we focus on DECLARE (Pesic and van der Aalst 2006; Montali et al. 2010; Ciccio and Montali 2022) – a pattern-based declarative process modelling language that relies on a predefined set of unary and binary templates, each of which can be represented as an LTLf formula. While there have been various analytic tasks studied for DECLARE and its data-aware extensions (Fionda and Greco 2018; Ciccio and Montali 2022; De Giacomo et al. 2022), to the best of our knowledge, there are no works addressing synthesis for declarative process specifications. Notice also that, since DECLARE patterns have been originally derived from a catalogue of relevant temporal properties in software engineering (Dwyer, Avrunin, and Corbett 1999), and are closely related to temporal patterns used in planning under trajectory constraints (Bacchus and Kabanza 2000; Gerevini et al. 2009), the findings of our work could be of interest for other communities outside the one of BPM.

In the next section we summarize our results on the synthesis problem formalization for DECLARE as well as related theoretical results reporting on the computational complexity of the realizability problem. We also highlight constructive optimizations that can be employed for boosting the synthesis problem solving in practice.

The Synthesis Approach

Differently from the case of standard synthesis, and similarly to the case of agents in AI (Zhu and De Giacomo 2022), also in BPM the execution of a process indeed comes with at least two parties and there is background knowledge on how the "uncontrollable" party (that is, Environment) operates. In particular, in BPM the external stakeholders participate to the execution of the process in a constrained way: when it is their turn, they decide which next action to trigger, with a controlled selection done among all and only the available actions offered by the information system supporting the enactment of the process (Dumas et al. 2018).

Reactive synthesis in such a setting comes as a natural problem, and, in the case of DECLARE, calls directly for a form of assume-guarantee synthesis, in the spirit of (Camacho, Bienvenu, and McIlraith 2018): Controller guarantees to enforce certain DECLARE constraints (called *guarantee*), under the assumption that Environment satisfies other DECLARE constraints (called *assumption*). Here, both sets of DECLARE constraints regulate how activities of respective parties must be executed. Then, the usual realizability problem takes as input two DECLARE models, one for the assumptions and the other for the guarantees to fulfill, and aims at finding a strategy (or a program) for Controller fulfilling all its constraints, regardless on how Environment behaves in the space of possibilities given by the assumption constraints. Such strategy can be seen as an *orchestration* indicating to the organization how to ensure that the guarantee is respected under the hypothesis that the external stakeholders behave by respecting the assumption. From the game-theoretic perspective, Environment looses the realizability game if it violates its assumptions.

Notice also that the use of the assume-guarantee approach is particularly important for DECLARE as the latter does not adopt the full expressiveness of LTLf, but focuses instead on conjunctively-related patterns of formulas.

We demonstrate that DECLARE realizability can be reduced to LTLf realizability by carefully manipulating specification formulas. This also means that it is easy to obtain a naïve algorithm for checking realizability of any DECLARE specification by simply using the traditional, doubly exponential time LTLf algorithm (see, e.g., (De Giacomo and Vardi 2015)). The same applies to reactive synthesis as well.

A natural question that arises at this point is whether the above algorithmic result could be improved. We give a positive answer to this question by obtaining a singly exponential time algorithm for DECLARE synthesis. This refined algorithm is based on the observation that, starting from pure past temporal formulas (that is, LTLf formulas using only past modalities like *yesterday*, *weak yesterday*, *since* and *triggers*), it is possible to build language-equivalent *deterministic* finite automata of singly exponential size (Chandra, Kozen, and Stockmeyer 1981; De Giacomo et al. 2021). For that we introduce a systematic encoding of all DECLARE patterns into linear-size pure-past formulas of LTLf. This is obtained via a procedure called *pastifiaction* (Cimatti et al. 2021; Maler, Nickovic, and Pnueli 2007, 2005) that transforms temporal formulas into equivalent pure past ones. The main advantage of such formulas is that there is no need for non-determinism (as the past "had already happened") and thus they can be transformed into DFAs with only a singly exponential blowup. To check the correctness of all the pastified DECLARE patterns w.r.t. their original formulation, we used the BLACK tool (Geatti et al. 2021; Geatti, Gigante, and Montanari 2021). As a by-product, our results also reveal the following fundamental property: DECLARE is a fragment of LTLf with a polynomial pastification algorithm (which, in general, is not true for full LTLf).

The above theoretical improvement already demonstrates that realizability for DECLARE specifications could be successfully addressed in practice. Given that since the organization of the SYNTCOMP (Jacobs et al. 2017), many optimized tools have been designed to solve reachability games over symbolic arenas, we propose a symbolic version of the above algorithm. To this end, we first study a novel translation from pure-past temporal formulas to linear-size *symbolic and deterministic* automata. In general, symbolic automata are preferred over explicit-state ones in such applications (*e.g.*, model checking or reactive synthesis), where the automaton sets of states and transitions are excessively large and cannot be stored in memory. The symbolic representation overcomes this issue by representing automata with Boolean formulas (McMillan 1993). On the average case, this representation can be exponentially more succinct than the classical one, and solving model checking or synthesis problems amounts thus to the reasoning over Boolean formulas representing the automata.

With this approach, the DFAs obtained from the pastified DECLARE patterns can be now represented with Boolean formulas that symbolically encode reachability games, which in turn paves the way towards a substantial improvement in performance thanks to the well-known practical benefits of the symbolic approach (Burch et al. 1992; McMillan 1993). It is also worth noting that the proposed symbolic DFA construction technique is much simpler and more efficient that the one studied in (Cimatti et al. 2021; Geatti 2022) as it does not use any counter bit and reduces the number of used state variables.

Conclusions and Future Directions

For the first time in the literature, we addressed the realizability and reactive synthesis problem for temporal specifications written in DECLARE. We gave a singly exponential time algorithm for DECLARE realizability, based on the technique of pastification, allowing to leverage the singly exponential DFA construction starting from pure past LTLf formulas. We also give a symbolic version of the previous algorithm, which has the potential of being applied in practice. We also believe that the proposed approach paves the way towards computational improvements in other automata-based tasks from the repertoire of techniques available for DECLARE (Ciccio and Montali 2022; De Giacomo et al. 2022), further fostering the cross-fertilization of AI and BPM.

References

Artale, A.; Mazzullo, A.; and Ozaki, A. 2019. Do You Need Infinite Time? In *Proc. of the 28th Int. Joint Conference on Artificial Intelligence (IJCAI-19)*. Macao, China, August 10-16: AAAI Press.

Bacchus, F.; and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2): 123–191.

Bansal, S.; Li, Y.; Tabajara, L. M.; and Vardi, M. Y. 2020. Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 9766–9774. AAAI Press.

Bloem, R.; Jobstmann, B.; Piterman, N.; Pnueli, A.; and Saar, Y. 2012. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences*, 78(3): 911–938.

Buchi, J. R.; and Landweber, L. H. 1990. Solving sequential conditions by finite-state strategies. In *The Collected Works of J. Richard Büchi*, 525–541. Springer.

Burch, J. R.; Clarke, E. M.; McMillan, K. L.; Dill, D. L.; and Hwang, L.-J. 1992. Symbolic model checking: 1020 states and beyond. *Information and computation*, 98(2): 142–170.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018. Finite LTL Synthesis as Planning. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M. T. J., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, 29–38. AAAI Press.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2018. Finite LTL Synthesis with Environment Assumptions and Quality Measures. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *Proceedings of KR'18*, 454–463. AAAI Press.

Chandra, A. K.; Kozen, D.; and Stockmeyer, L. J. 1981. Alternation. *J. ACM*, 28(1): 114–133.

Church, A. 1962. Logic, arithmetic, and automata. In *Proceedings of the international congress of mathematicians*, volume 1962, 23–35.

Ciccio, C. D.; and Montali, M. 2022. Declarative Process Specifications: Reasoning, Discovery, Monitoring. In van der Aalst, W. M. P.; and Carmona, J., eds., *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, 108–152. Springer.

Cimatti, A.; Geatti, L.; Gigante, N.; Montanari, A.; and Tonetta, S. 2021. Extended bounded response LTL: a new safety fragment for efficient reactive synthesis. *Formal Methods in System Design*, 1–49.

De Giacomo, G.; De Masellis, R.; Maggi, F. M.; and Montali, M. 2022. Monitoring Constraints and Metaconstraints with Temporal Logics on Finite Traces. *ACM Trans. Softw. Eng. Methodol.*, 31(4): 68:1–68:44.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proc. of the 28th National Conf. on Artificial Intelligence (AAAI-14)*, 1027–1033. AAAI Press.

De Giacomo, G.; Di Stasio, A.; Fuggitti, F.; and Rubin, S. 2021. Pure-past linear temporal and dynamic logic on finite traces. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4959–4965.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In Benton, J.; Lipovetzky, N.; Onaindia, E.; Smith, D. E.; and Srivastava, S., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, 128–136. AAAI Press.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 854–860. IJCAI/AAAI.

De Giacomo, G.; and Vardi, M. Y. 2015. Synthesis for LTL and LDL on Finite Traces. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 1558–1564. AAAI Press.

Dumas, M.; Rosa, M. L.; Mendling, J.; and Reijers, H. A. 2018. *Fundamentals of Business Process Management, Second Edition*. Springer.

Dwyer, M. B.; Avrunin, G. S.; and Corbett, J. C. 1999. Patterns in Property Specifications for Finite-State Verification. In Boehm, B. W.; Garlan, D.; and Kramer, J., eds., *Proceedings of the 1999 International Conference on Software Engineering, (ICSE'99)*, 411–420. ACM.

Fionda, V.; and Greco, G. 2018. LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner. *J. Artif. Intell. Res.*, 63: 557–623.

Geatti, L. 2022. *Temporal Logic Specifications: Expressiveness, Satisfiability And Realizability*. Phd thesis, University of Udine.

Geatti, L.; Gigante, N.; and Montanari, A. 2021. BLACK: A Fast, Flexible and Reliable LTL Satisfiability Checker. In Monica, D. D.; Pozzato, G. L.; and Scala, E., eds., *Proceedings of the 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, volume 2987 of *CEUR Workshop Proceedings*, 7–12. CEUR-WS.org.

Geatti, L.; Gigante, N.; Montanari, A.; and Venturato, G. 2021. Past Matters: Supporting LTL+Past in the BLACK Satisfiability Checker. In *Proceedings of the 28th International Symposium on Temporal Representation and Reasoning*.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.*, 173(5-6): 619–668.

Giacomo, G. D.; Favorito, M.; Li, J.; Vardi, M. Y.; Xiao, S.; and Zhu, S. 2022. LTLf Synthesis as AND-OR Graph

Search: Knowledge Compilation at Work. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2591–2598. ijcai.org.

Giacomo, G. D.; Masellis, R. D.; and Montali, M. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1027–1033. AAAI Press.

Giacomo, G. D.; Stasio, A. D.; Vardi, M. Y.; and Zhu, S. 2020. Two-Stage Technique for LTLf Synthesis Under LTL Assumptions. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, 304–314.

Harel, D.; and Pnueli, A. 1984. On the Development of Reactive Systems. In Apt, K. R., ed., *Logics and Models of Concurrent Systems - Conference proceedings, Colle-sur-Loup (near Nice), France, 8-19 October 1984*, volume 13 of *NATO ASI Series*, 477–498. Springer.

Jacobs, S.; Bloem, R.; Brenguier, R.; Ehlers, R.; Hell, T.; Könighofer, R.; Pérez, G. A.; Raskin, J.-F.; Ryzhyk, L.; Sankur, O.; et al. 2017. The first reactive synthesis competition (SYNTCOMP 2014). *International journal on software tools for technology transfer*, 19(3): 367–390.

Maggi, F. M.; Montali, M.; and Peñaloza, R. 2020. Temporal Logics Over Finite Traces with Uncertainty. In *Proc. of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, 10218–10225. AAAI Press.

Maler, O.; Nickovic, D.; and Pnueli, A. 2005. Real time temporal logic: Past, present, future. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 2–16. Springer.

Maler, O.; Nickovic, D.; and Pnueli, A. 2007. On synthesizing controllers from bounded-response properties. In *International Conference on Computer Aided Verification*, 95–107. Springer.

McMillan, K. L. 1993. Symbolic model checking. In *Symbolic Model Checking*, 25–60. Springer.

Montali, M.; Pesic, M.; van der Aalst, W. M. P.; Chesani, F.; Mello, P.; and Storari, S. 2010. Declarative specification and verification of service choreographiess. *ACM Trans. Web*, 4(1): 3:1–3:62.

Pesic, M.; and van der Aalst, W. M. P. 2006. A Declarative Approach for Flexible Business Processes Management. In Eder, J.; and Dustdar, S., eds., *Proceedings of BPM'06*, volume 4103 of *Lecture Notes in Computer Science*, 169–180. Springer.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57. IEEE.

Pnueli, A.; and Rosner, R. 1989a. On the Synthesis of a Reactive Module. In *Proceedings of POPL'89*, 179–190. ACM Press.

Pnueli, A.; and Rosner, R. 1989b. On the synthesis of an asynchronous reactive module. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 652–671. Springer.

Rosner, R. 1992. *Modular synthesis of reactive systems*. Ph.D. thesis, PhD thesis, Weizmann Institute of Science.

Xiao, S.; Li, J.; Zhu, S.; Shi, Y.; Pu, G.; and Vardi, M. Y. 2021. On-the-fly Synthesis for LTL over Finite Traces. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 6530–6537. AAAI Press.

Zhu, S.; and De Giacomo, G. 2022. Act for Your Duties but Maintain Your Rights. In Kern-Isberner, G.; Lakemeyer, G.; and Meyer, T., eds., *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022)*.

Zhu, S.; Giacomo, G. D.; Pu, G.; and Vardi, M. Y. 2020. LTLf Synthesis with Fairness and Stability Assumptions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 3088–3095. AAAI Press.

Zhu, S.; Pu, G.; and Vardi, M. Y. 2019. First-Order vs. Second-Order Encodings for \textsc ltl_f -to-Automata Translation. In Gopal, T. V.; and Watada, J., eds., *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13-16, 2019, Proceedings*, volume 11436 of *Lecture Notes in Computer Science*, 684–705. Springer.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. A Symbolic Approach to Safety LTL Synthesis. In Strichman, O.; and Tzoref-Brill, R., eds., *Proceedings of the 13th International Haifa Verification Conference*, volume 10629 of *Lecture Notes in Computer Science*, 147–162. Springer.