Synthesis from Temporal Specifications: New Applications in Robotics and Model-Driven Development

Nir Piterman

University of Leicester

Synthesis from temporal specifications is the automatic production of adaptable plans (or input enabled programs) from high level descriptions. The assumption underlying this form of synthesis is that we have two interacting reactive agents. The first agent is the system for which the plan / program is being designed. The second agent is the environment with which the system interacts. The exact mode of interaction and the knowledge available to each of the agents depends on the application domain. The high level description of the plan is usually given in some form of temporal logic, where we often distinguish between assumptions and guarantees. As we do not expect the system to function correctly in arbitrary environments, the assumptions detail what the system expects from the environment. The guarantees are what the system is expected to fulfill in such environments. Our algorithms then produce a plan that interacts with the environment and reacts to it so that the tasks assigned to the plan are fulfilled. By definition, the plan is reacting to the moves of the environment and tries to adapt itself to the current condition (as a function of that interaction).

Technically, the interaction between the system and its environment is modeled as a two-player game, where system choices correspond to the execution of the plan and environment choices correspond to the behavior of the environment. The specifications, i.e., the assumptions on the environment and the guarantees of the system, are translated to the winning conditions in the game: the system has to be able to resolve its choices in such a way that it satisfies the specification. The way the system resolves its choices, called *strategy*, is then translated to a design that satisfies the specification. Verifying that such a strategy exists and computing the strategy is referred to as “solving the game”. Different types of games arise depending on the exact conditions of the interaction between the agents, and depending on the winning conditions. In order to make synthesis useful we have to come up with algorithms that work well for the games that arise from interesting applications.

The theoretical framework for synthesis from temporal specifications has been known for many years. The question of decidability of this form of synthesis was raised by Church in the late 50’s [8]. Independently, Rabin [29] and Büchi and Landweber [7] suggested tree automata and two-player games as a way to reason about the interaction between the program and its environment. These solutions concentrated on decidability and were not concerned with practicality. Pnueli and Rosner cast this question in a modern setting and proved that synthesis from linear-time temporal logic (LTL) specifications is 2EXPTIME-complete [28]. Indeed, this is the framework considered here.

The solution of Pnueli and Rosner called for the translation of the specification to a deterministic Rabin automaton over infinite words [31]. Integrating this automaton with the approach of Rabin produced a Rabin tree automaton accepting winning strategies. Checking emptiness of this automaton corresponds to deciding whether the specification is *realizable*. Finding a tree accepted by this automaton corresponds to extracting a strategy. The two components of this solution proved very hard to implement. Determinization of automata on infinite words proved complicated to implement [18, 1]. To the best of our knowledge, emptiness of Rabin automata (equivalently solution of Rabin games) was never implemented [13, 28, 26]. Improvements to determinization [25] are still challenging to implement effectively [34]. They lead to the slightly simpler parity automata / games, for which no efficient solution is known [17, 15, 32].

These difficulties led researchers to suggest two ways to bypass the two complicated parts of this approach. One approach is to avoid determinization and reduce synthesis to safety games [24, 14, 33]. This approach has been implemented in various tools [16, 12, 5]. The second approach, the one advocated here, is to restrict attention to a subset of LTL that can be solved more efficiently [27, 4].

Specifically, we consider LTL formulas over Boolean variables partitioned to sets of *inputs* and *outputs*, $\mathcal{X}$ and $\mathcal{Y}$, respectively. Then, the specification has the format $\varphi_e \rightarrow \varphi_s$, where φ_e is a conjunction of assumptions on the behavior of the environment and φ_s is a conjunction of guarantees of the system. Both φ_e and φ_s are restricted to the form $\psi_i^a \wedge \Box\, \rho_t^a \wedge \bigwedge_{i \in I_g^a} \Box \Diamond J_i^a$, for $a \in \{e, s\}$, where the components of φ_a take the following form.

- ψ_i^e is a Boolean formula over $\mathcal{X}$ and ψ_i^s is a Boolean formula over $\mathcal{X} \cup \mathcal{Y}$.
- ρ_t^e is a Boolean formula over $\mathcal{X} \cup \mathcal{Y}$ and $\bigcirc\, \mathcal{X}$ and ρ_t^s is a Boolean formula over $\mathcal{X} \cup \mathcal{Y}$ and $\bigcirc\, \mathcal{X} \cup \bigcirc\, \mathcal{Y}$. That is, ρ_t^e is allowed to relate to the next values of input variables and ρ_t^s is allowed to relate to the next values of both input and output variables..
- J_i^e is a Boolean formula over $\mathcal{X} \cup \mathcal{Y}$.

That is, the specification takes the following format:

$$\left(\psi_i^e \wedge \Box\, \rho_t^e \wedge \bigwedge_{i \in I_g^e} \Box \Diamond J_i^e \right) \rightarrow \left(\psi_i^s \wedge \Box\, \rho_t^s \wedge \bigwedge_{i \in I_g^s} \Box \Diamond J_i^s \right)$$

Intuitively, this formula allows the system to update its initial assignment to output variables based on some assumption on the assignment to the input variables; it allows the system to update output variables based on the way the environment updates the input variables; and it allows the system to fulfill some liveness requirements based on the environment fulfilling its own liveness requirements.[1] We argue that this form of specifications arise in practice and

---

[1] We note that presentation of the specification in the form of such an implication depends on the ability of the environment to fulfil its assumptions [19, 4].

are sufficient to specify many interesting designs. Furthermore, we show how to implement the solution to the synthesis problem arising from such specifications using BDDs.

This approach has been adopted by some practitioners and led to applications of synthesis in hardware design [2, 3] robot-controller planning [9, 20, 21, 35, 37, 36], and user programming [22, 23]. Adapting our solution to be used in the context of robot-controller required to consider how to combine the discrete controller produced by our approach with continuous controllers for various parts of the robot [30]. Recently, we have adapted this approach to applications in model-driven development [10, 11, 6]. This required us to adjust setting to that of games defined by labeled-transition systems, winning conditions defined by fluent linear-temporal logic, and to enumerative representation of games.

Here we will survey the theoretical solution to synthesis proposed by Pnueli and Rosner and some of the difficulties in applying it in practice. We will then present our approach and some of the applications it was used for. We will also cover some of the issues arising from adaptation of our approach to the usage by practitioners in robotics and model-driven development.

Acknowledgements

The work surveyed here is based on joint work with (mostly) Roderick Bloem, Victor Braberman, Nicolas D'Ippolito, Barbara Jobstmann, Hadas Kress-Gazit, Amir Pnueli, Vasu Raman, Yaniv Sa'ar, and Sebastian Uchitel. References to our joint work are mentioned in the paper. I am grateful to them for the great pleasure in working with them on these results.

References

1. C. Althoff, W. Thomas, and N. Wallmeier. Observations on determinization of Büchi automata. In *Proc. 10th Int. Conf. on the Implementation and Application of Automata*, volume 3845 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2005.
2. R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Automatic hardware synthesis from specifications: A case study. In *Design Automation and Test in Europe*, pages 1188–1193, 2007.
3. R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Specify, compile, run: Hardware from PSL. In *6th International Workshop on Compiler Optimization Meets Compiler Verification*, volume 190 of *Electronic Notes in Computer Science*, pages 3–16, 2007.
4. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and Systems Science*, 78(3):911–938, 2012.
5. A. Bohy, V. Bruyère, E. Filiot, N. Jin, and J.-F. Raskin. Acacia+, a tool for LTL synthesis. In *24th International Conference on Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 652–657. Springer, 2012.

6. V. Braberman, N. D'Ippolito, N. Piterman, D. Sykes, and S. Uchitel. Controller synthesis: From modelling to enactment. In *35th International Conference on Software Engineering*, San Francisco, USA, 2013.
7. J. Büchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Trans. AMS*, 138:295–311, 1969.
8. A. Church. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume I, pages 3–50, Cornell University, Ithaca, 1957.
9. D. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. Pappas. Valet parking without a valet. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 572–577. IEEE, 2007.
10. N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesis of live behavior models for fallible domains. In *33rd International Conference on Software Engineering*. ACM, ACM press, 2011.
11. N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesising non-anomalous event-based controllers for liveness goals. *Transactions on Software Engineering and Methodology*, 22(1), 2012.
12. R. Ehlers. Symbolic bounded synthesis. *Formal Methods in System Design*, 40(2):232–262, 2012.
13. E. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, 1988.
14. T. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.
15. M. J. J. Voge. A discrete strategy improvement algorithm for solving parity games. In *Proc 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2000.
16. B. Jobstmann and R. Bloem. Game-based and simulation-based improvements for ltl synthesis. In *3nd Workshop on Games in Design and Verification*, 2006.
17. M. Jurdziński. Deciding the winner in parity games is in up $\cap$ co-up. *Information Processing Letters*, 68(3):119–124, 1998.
18. J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. In *Proc. 10th Int. Conf. on Implementation and Application of Automata*, Lecture Notes in Computer Science. Springer, 2005.
19. U. Klein and A. Pnueli. Revisiting synthesis of gr(1) specifications. In *6th International Haifa Verification Conference*, volume 6504 of *Lecture Notes in Computer Science*, pages 161–181. Springer, 2010.
20. H. Kress-Gazit, G. Fainekos, and G. Pappas. From structured english to robot motion. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2722. IEEE, 2007.
21. H. Kress-Gazit, G. Fainekos, and G. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3116–3121. IEEE, 2007.
22. H. Kugler, C. Plock, and A. Pnueli. Controller synthesis from lsc requirements. In *Proc. Fundamental Approaches to Software Engineering*, volume 5503 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2009.
23. H. Kugler and I. Segall. Compositional synthesis of reactive systems from live sequence chart specifications. In *Proc. 15th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 5505 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2009.

24. O. Kupferman, N. Piterman, and M. Vardi. Safraless compositional synthesis. In *Proc 18th Int. Conf. on Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006.
25. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):e5, 2007.
26. N. Piterman and A. Pnueli. Faster solution of Rabin and Streett games. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 275–284. IEEE, IEEE Computer Society Press, 2006.
27. N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. In *Proc. 7th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 364–380. Springer, 2006.
28. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.
29. M. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
30. V. Raman, N. Piterman, and H. Kress-Gazit. Provably correct continuous control for high-level robot behaviors with actions of arbitrary execution durations. In *IEEE International Conference on Robotics and Automation*. IEEE, IEEE Computer Society Press, 2013.
31. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
32. S. Schewe. Solving parity games in big steps. In *27th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4855 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2007.
33. S. Schewe and B. Finkbeiner. Bounded synthesis. In *4th Int. Symp. on Automated Technology for Verification and Analysis*, volume 4218 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2006.
34. M.-H. Tsai, S. Fogarty, M. Vardi, and Y.-K. Tsay. State of büchi complementation. In *15th International Conference on Implementation and Application of Automata*, volume 6482 of *Lecture Notes in Computer Science*, pages 261–271. Springer, 2010.
35. T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *IEEE Conference on Decision and Control*, pages 5997–6004. IEEE Computer Society Press, 2009.
36. T. Wongpiromsarn, U. Topcu, and R. M. Murray. Automatic synthesis of robust embedded control software. In *AAAI Spring Symposium on Embedded Reasoning: Intelligence in Embedded Systems*, 2010.
37. T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer, 2010.