Recent Advances in LTL Realizability and Synthesis as Planning

Alberto Camacho
University of Toronto, Canada
acamacho@cs.toronto.edu

1 Introduction

Automatic synthesis of software from specification is a classic problem in computer science that dates back to Church in 1957. Synthesis is a hard problem that has been well-studied, and for which no efficient solution exists in the general case. In the context of constructing strategies for reactive systems, Pnueli and Rosner (1989) proposed *Linear Temporal Logic* (LTL) synthesis where the specification is expressed in LTL. In the last decade we have seen significant algorithmic advances in LTL synthesis, and the field is gathering signifficant traction. So-called *bounded synthesis* techniques transform the problem into a *game*, and limit the search for solutions to spaces of bounded size that are more tractable, in practice (Kupferman and Vardi 2005; Schewe and Finkbeiner 2007). Bounded synthesis was a major breakthrough in the development of practical algorithms. Modern tools approach the problem as bounded synthesis, and solve the resulting games using different technology such as SAT (e.g. (Bohy et al. 2012)), SMT (e.g. (Faymonville, Finkbeiner, and Tentrup 2017)), and BDDs (e.g. LtlSynt). We explore the use of automated planning.

This document summarizes some of the recent work by our research group. We study the synthesis problem of LTL specifications for programs that run in perpetuity, and for programs that terminate in finite time. From a theoretical perspective, our work makes the following contributions:

- formalize the correspondence between synthesis and automated planning, as exemplified in (Camacho et al. 2018b; 2018a; 2018d)
- define and generate certificates of unrealizability for programs that terminate (Camacho et al. 2018a)
- introduce novel techniques for determining LTL realizability via reachability games (Camacho et al. 2018d)
- examine the critical role of environment assumptions in the generation of programs that terminate (Camacho, Bienvenu, and McIlraith 2018)
- provide novel quality measures and algorithms for the synthesis of *optimal* programs that terminate (Camacho, Bienvenu, and McIlraith 2018)

Our experiments suggest that planning can be an efficient technology to approach to synthesis. Most of our techniques are implemented in a a web service and API tool for synthesis, that we name SynKit (Camacho et al. 2018e), and that aims at making synthesis more accessible to the masses.

1.1 What is LTL Realizability and Synthesis?

LTL is a modal logic defined over a set of atomic propositions $p \in AP$ with standard logical connectives and temporal operators *next* ($\bigcirc$) and *until* ($\mathcal{U}$).

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

Intuitively, LTL formula $\bigcirc\alpha$ denotes that α has to hold in the next time step, and $\alpha\mathcal{U}\beta$ denotes that α has to hold until β holds. The truth of an LTL formula is typically evaluated over infinite traces. Noteworthy, LTL formulae can be also interpreted over *finite* traces (e.g. (Bacchus and Kabanza 1998; Baier and McIlraith 2006; De Giacomo and Vardi 2013)). This variant is more recently referred to as LTL_f, which we also use to disambiguate with the infinite semantics.

LTL (resp. LTL_f) is a compelling language to express the properties of the programs that we want to synthesize, and that have to run in perpetuity (resp. terminate in finite time).

Synthesis of programs that run in perpetuity Following the notation in (Camacho et al. 2018d), an LTL specification is a tuple $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle_s$, where $\mathcal{X}$ and $\mathcal{Y}$ are disjoint sets of variables and φ is an LTL formula over $\mathcal{X} \cup \mathcal{Y}$. LTL specifications are usually interpreted as two-player games, where the *environment* player controls $\mathcal{X}$ and the *agent* player controls $\mathcal{Y}$. In each turn, players select a subset of the variables they control. A *play* is an infinite sequence $w = (x_1 \cup y_1)(x_2 \cup y_2) \cdots$ of subsets of $\mathcal{X} \cup \mathcal{Y}$. The play is winning for the agent if w satisfies φ. LTL realizability consists of determining if the agent player has a winning strategy for the game, and LTL synthesis is the problem of computing one such strategy. The order of turn taking is important, and is indicated by the *semantics*, s. If the agent plays first, then s = "Moore"; otherwise, s = "Mealy" (e.g. (Ehlers 2011))

Synthesis of programs that terminate LTL_f realizability and synthesis are defined as in the infinite case, with the exception that φ is an LTL_f formula (De Giacomo and Vardi 2015). Likewise, LTL_f specifications can be interpreted as two-player games. This time, the objective is for the agent to find a strategy that satisfies the LTL_f formula in a *finite* number of turns. LTL_f synthesis captures many interesting and important problems that have finite duration, including typical planning problems and problems involving business processes.

2 Programs that Run in Perpetuity

LTL realizability is typically determined by solving synthesis in a *dual* game, where the role of the system and environment players is interchanged. In recent work, we revisited the role of duality in LTL realizability and synthesis (Camacho et al. 2018d). We distinguish two types of duality: the duality between synthesis and automata games, and the duality with respect to the Mealy and Moore semantics. By carefully exploiting duality, we provide a unified view of synthesis and games, and introduce novel *bounded realizability* methods via reductions to *reachability* games – that is, games where a "good thing" has to occur in finite time.

3 Programs that Terminate

Existing approaches to LTL_f synthesis transform LTL_f into *deterministic* finite-state automata (DFA) and reduce the synthesis problem to a DFA game (e.g. (Zhu et al. 2017)). Unfortunately, the DFA transformation is worst-case double-exponential in the size of the formula, presenting a computational bottleneck. In contrast, our approach exploits *non-deterministic* automata. We leverage our approach not only for strategy generation but also to generate certificates of unrealizability – the first such method for LTL_f.

In further work, we examine the critical role of environment assumptions in the synthesis of terminating programs (Camacho, Bienvenu, and McIlraith 2018). We argue that in many applications, the existence of a program that satisfies user intent is predicated on some assumption on the environment behavior. While this is also true for LTL synthesis, we notice that incorporating and handling environment assumptions is significantly more intricate for LTL_f synthesis.

Finally, we explore different quality measures and introduce novel techniques to compute optimal solutions to LTL_f specifications (Camacho, Bienvenu, and McIlraith 2018).

4 From Theory to Practice

We implemented our algorithms for LTL and LTL_f realizability and synthesis via automated planning in a tool that we named *SynKit* (Camacho et al. 2018e). *SynKit* is the first tool that computes certificates of unrealizability for LTL_f specifications, and is the first algorithmic approach to LTL realizability via reachability games. Our experiments show that automated planning is a promising approach to synthesis, with competitive performance relative to state of the art.

SynKit is accessible as a web service and API. Our purpose is to make synthesis technology more accessible to the masses, given that there is a limited number of existing tools.

5 Related Work

Some of our algorithmic techniques were inspired by previous work in planning with LTL_f goals (e.g. (Baier and McIlraith 2006; Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017b)). Likewise, our novel techniques can be applied into other sequential decision-making problems. In recent work we developed techniques to solve *Non-Markovian Reward Decision Processes* (NMRDPs) with LTL_f rewards (Camacho et al. 2017a; 2018c). Similar work has been recently pursued by (Brafman, De Giacomo, and Patrizi 2018).

References

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* 22(1-2):5–27.

Baier, J. A., and McIlraith, S. A. 2006. Planning with temporally extended goals using heuristic search. In *ICAPS*, 342–345.

Bohy, A.; Bruyère, V.; Filiot, E.; Jin, N.; and Raskin, J. 2012. Acacia+, a tool for LTL synthesis. In *CAV*, 652–657.

Brafman, R. I.; De Giacomo, G.; and Patrizi, F. 2018. LTLf/LDLf non-markovian rewards. In *AAAI*, 1771–1778.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2017a. Non-markovian rewards expressed in LTL: Guiding search via reward shaping. In *SOCS*, 159–160.

Camacho, A.; Triantafillou, E.; Muise, C.; Baier, J. A.; and McIlraith, S. A. 2017b. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, 3716–3724.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018a. Finite LTL synthesis as planning. In *ICAPS*, 29–38.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018b. Synthesizing controllers: On the correspondence between LTL synthesis and non-deterministic planning. In *Advances in Artificial Intelligence - Canadian AI*, 45–59.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2018c. Non-markovian rewards expressed in LTL: Guiding search via reward shaping (extended version). In *GoalsRL Workshop*.

Camacho, A.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2018d. LTL realizability via safety and reachability games. In *IJCAI*, 4683–4691.

Camacho, A.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2018e. SynKit: LTL synthesis as a service. In *IJCAI*, 5817–5819.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2018. Finite LTL synthesis with environment assumptions and quality measures. In *KR*. To appear.

Church, A. 1957. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic, Cornell University 1957* 1:3–50.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 854–860.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 1558–1564.

Ehlers, R. 2011. Experimental aspects of synthesis. In *IWIGP*, 1–16.

Faymonville, P.; Finkbeiner, B.; and Tentrup, L. 2017. Bosy: An experimentation framework for bounded synthesis. In *CAV*, 325–332.

Kupferman, O., and Vardi, M. Y. 2005. Safraless decision procedures. In *FOCS*, 531–542.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *IJCAI*, 2343–2349.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*, 179–190.

Schewe, S., and Finkbeiner, B. 2007. Bounded synthesis. In *ATVA*, 474–488.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTLf synthesis. In *IJCAI*, 1362–1369.