Poster: Adaptive Strategies for rLTL Games

Satya Prakash Nayak
Chennai Mathematical Institute, India

Daniel Neider
MPI-SWS, Germany

Martin Zimmermann
University of Liverpool, UK

ABSTRACT

We consider the problem of synthesizing the most robust controllers using the Abstraction-Based Controller Design (ABCD). First, we perform a finite-state abstraction of the continuous dynamic system. We then synthesize a most robust control strategy in the finite space by formulating it as a two-player game. Finally, we refine the strategy to a controller for the original problem. To preserve robustness, we consider the specifications for the controllers to be expressed in Robust Linear Temporal Logic (rLTL), which allows the reasoning about how robust the specification is. However, the current algorithms for rLTL synthesis do not compute optimally robust controllers. It only considers the worst-case analysis for reactive synthesis. Hence, we develop two new notions of adaptive strategies. One is *Weakly Adaptive strategy*, which, in response to the opponent's bad choices, adaptively changes the degree of satisfaction we want to achieve to ensure the optimality w.r.t. the current stage. The second one is *Strongly adaptive strategy*, which is weakly adaptive that also maximizes the chances of the opponent making a bad choice. We show that the computability problem for both the strategies is not harder than the classical one and can be solved in doubly-exponential time.

CCS CONCEPTS

• **Theory of computation → Modal and temporal logics**; **Logic and verification**.

ACM Reference Format:
Satya Prakash Nayak, Daniel Neider, and Martin Zimmermann. 2021. Poster: Adaptive Strategies for rLTL Games. In *24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3447928.3457210

1 INTRODUCTION

Nowadays, formal methods are being used extensively to specifying control system requirements. One way to construct controllers for continuous dynamic systems with temporal specifications is Abstraction-Based Controller Design (ABCD) [1]. The ABCD principle first computes a finite and discrete-time abstraction of the continuous dynamic system. Then using reactive synthesis, it computes a discrete controller for the finite system with the temporal specification. And finally, it refines the discrete controller to a controller for the original continuous system. This methodology has recently been implemented in various algorithms and tools.

HSCC '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8339-4/21/05.
https://doi.org/10.1145/3447928.3457210

This paper addresses the problem of synthesizing the controller with specifications expressed in Robust Linear Temporal Logic (rLTL) [4]. Robust LTL was introduced by Tabuada and Neider [4] to capture the concept of robustness in temporal logics. It was observed that the difference between "minor" and "major" violations of a formula cannot be distinguished in a two-valued semantics. For example, consider the formula $\varphi = \Box p$, which demands that p holds at all positions of a word. Clearly, φ is violated even if p does not hold at only a single position, which is a very minor violation. However, the two-valued semantics of LTL does not distinguish between this case and the case where p does not hold at any position, which is a major violation. To distinguish these various degrees of violations, rLTL adopts a 5-valued semantics. The set of truth values for rLTL is $\mathbb{B}_4 = \{1111, 0111, 0011, 0001, 0000\}$ and the values are ordered naturally. Intuitively, 1111 corresponds to true, and the rest to different shades of false. For the above example, the robust version of the formula φ is written as $\boxdot p$, then, the five truth values distinguish the various degree of violations of the property:

- The value is 1111 if p holds at all positions (no violation).
- The value is 0111 if p holds eventually always, i.e., p holds at all but finitely many positions.
- The value is 0011 if p holds at infinitely many positions.
- The value is 0001 if p holds at finitely many positions.
- The value is 0000 if p does not hold at any position.

We focus on the problem of synthesizing the most robust controllers, i.e., the optimal controllers w.r.t. the natural ordering on $\mathbb{B}_4$ in a finite state space. Such problems can be formulated as finite-state graph-based games between the environment and the controller, called rLTL games [4]. However, the classical controllers computed by Tabuada and Neider are not optimal. It considers the environment to be antagonistic, which is not very realistic. In order to solve this inefficiency, we introduce two new notions of adaptive strategies: weakly adaptive strategy and strongly adaptive strategy. First one is a strategy that adapts its moves to ensure the optimality once the environment has made a bad move. Second one is a stronger version of first one that also maximizes the chances of the environment making bad moves. We show that both the strategies can be computed (if exist) in doubly-exponential time, and hence are not harder than the classical ones.

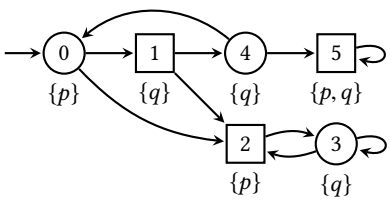

Figure 1: Game graph for Example 1

2 COMPUTING ADAPTIVE STRATEGIES

In this section, we motivate our work with an example of a game. We first show what a classical strategy would do in that game and why that is not optimal. Moreover, we present a weakly adaptive strategy and describe why it is better than the classical one. We also present a strongly adaptive strategy and describe its importance and existence.

2.1 Motivating Example

Consider a game played between two players: Player C (controller) and Player E (environment), as shown in Figure 1. Player E's vertices are shown as squares and Player C's vertices are shown as circles. Each vertex is labeled by a set of propositions, e.g., vertex 5 is labeled by propositions p and q. Suppose a token is initially placed at the vertex 0. At any stage, if the token is in a vertex of Player i, then he has to move the token to a neighboring vertex along an edge. An infinite play is an infinite path in the graph starting from 0, whose labels induce an infinite word consisting of sets of propositions, e.g., the play 012323 . . . induces to word $\{p\}\{q\}\{p\}\{q\}\{p\}\{q\}\ldots$ Suppose the rLTL specification is $\boxdot p$ for Player C, which means he wants p to hold at all positions of (the word induced by) the infinite play (which is not possible in this game). Then he would prefer a play where p holds eventually always (e.g., 01455 . . .) over a play where p holds at infinitely many positions (e.g., 012323 . . .) Similarly, he would prefer a play where p holds at infinitely many positions over a play where p holds at finitely many positions (e.g., 01233 . . .). Hence, Player C's objective is to maximize the value of $\boxdot p$ on the play. Since reactive synthesis performs a worst-case analysis, the environment is considered antagonistic, and the objective of Player E is to minimize the value on any play.

2.2 Weakly Adaptive Strategy

A *strategy* for Player i is a function σ, which assigns to each possible finite path ending in a vertex of Player i to a neighbour of that vertex. Intuitively, it prescribes the next move of Player i depending on the finite play played thus far. Considering Player E plays his best moves, the best possible scenario for Player C in the above example is to enforce a play where p holds at infinitely many positions. As the classical problem only considers the worst-case analysis, a classical strategy for Player C is to try to visit the vertex 2 infinitely often. That can be done by moving the token along one of the following edges every time the token reaches his vertices: $\{0 \to 1; 3 \to 2; 4 \to 1\}$. As we can see that, if Player E makes a bad move by moving along $1 \to 4$, then Player C can force the play to eventually just stay at the vertex 5, and hence, p holds eventually always. However, the above classical strategy for Player C moves it back to the vertex 0 from which p might not hold eventually always. Therefore, a better strategy for Player C is to move along $4 \to 5$ if the token reaches the vertex 4 to get a play where p holds eventually always; otherwise, try to get a play where p holds at infinitely many positions as earlier by moving along $0 \to 1$ and then $3 \to 2$ repeatedly. We call such a strategy *weakly adaptive*. Intuitively, it adapts its moves to achieve the best possible outcome after each bad move of the environment.

It can be shown that a weakly adaptive strategy for a player in an rLTL game can be computed in doubly-exponential time by reducing the problem to parity games [2]. As we know that the classical synthesis problems for the controllers for an rLTL (even for LTL) specification also take doubly-exponential time [4], we conclude that computing weakly adaptive strategies is not harder than the standard ones.

2.3 Strongly Adaptive Strategy

Another weakly adaptive strategy for Player C is to move along $0 \to 2$ directly in his first move and then moving along $3 \to 2$ every time. Then the token can never reach the vertex 4. However, it also means that there cannot be a play where p holds eventually always; whereas if Player C moves along $0 \to 1$, there is a chance of getting such plays. Therefore, the earlier strategy of moving the token to 1 is definitely better. Therefore, we also consider another type of strategy, which is weakly adaptive, as well as it maximizes the chances of the environment making a bad move. We call such a strategy *strongly adaptive*. However, such a strategy may not even exist for some cases, whereas it is easy to see that a weakly adaptive strategy always exists. We illustrate this in the following example.

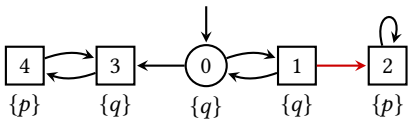

Figure 2: Game graph for Example 2

Consider another game with the same objectives played in the game graph shown in Figure 2. Suppose the token is initially placed at the vertex 0. If Player E plays his best moves, then the best possible play Player C can enforce is the one where p holds at infinitely many positions (e.g., a play with suffix 03434 . . .). So unless Player E makes a bad move by moving along $1 \to 2$, any weakly adaptive strategy for Player C will eventually make him move the token to 3. But if Player C moves along $0 \to 1$, then there is a chance of Player E making a bad move of $1 \to 2$, and hence the token stays at the vertex 2, inducing a play where p holds eventually always. So, if σ_k is a strategy for Player C, which makes him move along $0 \to 1$ the first k times it reaches 0 and then moves to 3; then σ_{k+1} is always a better strategy than σ_k. Hence, no strongly adaptive strategy exists.

It can be shown that a strongly adaptive strategy, if one exists at all, can also be synthesized in doubly-exponential time by a reduction to a series of parity games [2] and obliging games [3]. Hence, computing strongly adaptive strategy is also not harder than the classical ones.

REFERENCES

[1] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. 2017. *Formal methods for discrete-time dynamical systems.* Vol. 15. Springer.
[2] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. 2017. Deciding Parity Games in Quasipolynomial Time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (Montreal, Canada) *(STOC 2017)*. Association for Computing Machinery, New York, NY, USA, 252–263. https://doi.org/10.1145/3055399.3055409
[3] Krishnendu Chatterjee, Florian Horn, and Christof Löding. 2010. Obliging Games. In *CONCUR 2010 - Concurrency Theory*, Paul Gastin and François Laroussinie (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 284–296.
[4] Paulo Tabuada and Daniel Neider. 2016. Robust Linear Temporal Logic. In *CSL (LIPIcs, Vol. 62)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 10:1–10:21.