Robustness-by-Construction Synthesis: Adapting to the Environment at Runtime

Satya Prakash Nayak[1][0000−0002−4407−8681], Daniel Neider[2][0000−0001−9276−6342], and Martin Zimmermann[3][0000−0002−8038−2453]

[1] Max Planck Institute for Software Systems, Kaiserslautern, Germany
sanayak@mpi-sws.org
[2] Safety and Explainability of Learning Systems Group
Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany
daniel.neider@uni-oldenburg.de
[3] Aalborg University, Aalborg, Denmark
mzi@cs.aau.dk

Abstract. While most of the current synthesis algorithms only focus on correctness-by-construction, ensuring robustness has remained a challenge. Hence, in this paper, we address the robust-by-construction synthesis problem by considering the specifications to be expressed by a robust version of Linear Temporal Logic (LTL), called robust LTL (rLTL). rLTL has a many-valued semantics to capture different degrees of satisfaction of a specification, i.e., satisfaction is a quantitative notion.
We argue that the current algorithms for rLTL synthesis do not compute optimal strategies in a non-antagonistic setting. So, a natural question is whether there is a way of satisfying the specification "better" if the environment is indeed not antagonistic. We address this question by developing two new notions of strategies. The first notion is that of adaptive strategies, which, in response to the opponent's non-antagonistic moves, maximize the degree of satisfaction. The idea is to monitor non-optimal moves of the opponent at runtime using multiple parity automata and adaptively change the system strategy to ensure optimality. The second notion is that of strongly adaptive strategies, which is a further refinement of the first notion. These strategies also maximize the opportunities for the opponent to make non-optimal moves. We show that computing such strategies for rLTL specifications is not harder than the standard synthesis problem, e.g., computing strategies with LTL specifications, and takes doubly-exponential time.

1 Introduction

Formal methods have focused on the paradigm of correctness-by-construction, i.e., ensuring that systems are guaranteed to meet their design specifications. While correctness is necessary, it has widely been acknowledged that this property alone is insufficient for a good design when a reactive system interacts with an ever-changing, uncontrolled environment. To illustrate this point, consider a typical correctness specification $\varphi \Rightarrow \psi$ of a reactive system, where φ is an environment assumption and ψ the system's desired guarantee. Thus, if the environment violates φ, the entire implication becomes vacuously true, regardless of whether the system satisfies ψ. In other words, if the assumption about the environment is violated, the system may behave arbitrarily. This behavior is clearly undesirable as modeling any reasonably complex environment accurately and exhaustively is exceptionally challenging, if not impossible.

The example above shows that reactive systems must not only be correct but should also be *robust* to unexpected environment behavior. The notion of robustness we use in this paper is inspired by concepts from control theory [20,32,33,35] and requires that deviations from the environment assumptions result in at most proportional violations of the system guarantee. More precisely, "minor" violations of the environment assumption should only cause "minor" violations of the system guarantee, while "major" violations of the environment assumption allow for "major" violations of the system guarantee.

To capture different degrees of violation (or satisfaction) of a specification, we rely on a many-valued extension of Linear Temporal Logic (LTL) [28], named *robust Linear Temporal Logic (rLTL)*, which has recently been introduced by Tabuada and Neider [36]. The basic idea of this logic can best be illustrated by considering the prototypical environment assumption $\varphi \coloneqq \Box p$ ("always p"), which demands that the environment ensures that an atomic proposition p holds at every step during its interaction with the system. Clearly, φ is violated even if p does not hold at a single step, which is a "minor" violation. However, the classical Boolean semantics of LTL cannot distinguish between this case and the case where p does not hold at any position, which is a "major" violation. To distinguish these (and more) degrees of violations, rLTL adopts a five-valued semantics with truth values $\mathbb{B}_4 = \{1111, 0111, 0011, 0001, 0000\}$. The set $\mathbb{B}_4$ is ordered according to $1111 > 0111 > 0011 > 0001 > 0000$, where 1111 is interpreted as *true* and all other values as increasing shades of *false*. In case of the formula φ, for instance, the interpretation of these five truth values is as follows: φ evaluates to 1111 if the environment ensures p at every step of the interaction, φ evaluates to 0111 if p holds almost always, φ evaluates to 0011 if p holds infinitely often, φ evaluates to 0001 if p holds at least once, and φ evaluates to 0000 if p never holds. The semantics of rLTL is then set up so that $\varphi \Rightarrow \psi$ evaluates to 1111 if any violation of the environment assumption φ causes at most a proportional violation of the system guarantee ψ (i.e., if φ evaluates to truth value $b \in \mathbb{B}_4$, then ψ must evaluate to a truth value $b' \geq b$).

Here, we are interested in the synthesis problem for rLTL specifications. As usual, we model such a synthesis problem as an infinite-duration two-player game. Since we study rLTL synthesis, we consider games with rLTL winning conditions, so-called rLTL games.

rLTL games with a Boolean notion of winning strategy for the system player have already been studied by Tabuada and Neider [36]. In their setting, the objective for the system player is as follows: given a truth value $b \in \mathbb{B}_4$, he must react to the actions of the environment player in such a way that the specification is satisfied with a value of at least b. As for ω-regular games, a winning strategy for the system player can immediately be implemented in hardware or software. This implementation then results in a reactive system that is guaranteed to satisfy the given specification with at least a given truth value $b \in \mathbb{B}_4$, regardless of how the environment acts.

While rLTL games provide an elegant approach to robustness-by-construction synthesis, the Boolean notion of winning strategies that Tabuada and Neider adopt has a substantial drawback: it does not incentivize the system player to satisfy the specification with a value better than b, even if the environment player allows this. Of course, one can (and should) statically search for the largest $b \in \mathbb{B}_4$ such that the system player can win the game. However, this traditional worst-case view does not account for many practical situations where the environment is not antagonistic, e.g., in the presence of intermittent disturbances or noise, or when the environment cannot be modeled entirely [15,16,25,26,37]. In such situations, the system player should exploit the environment's "bad" moves, i.e., actions that permit the system player to achieve a value greater than b, and adapt its strategy at runtime.

We present two novel synthesis algorithms for rLTL specifications that ensure that the resulting systems are *robust by construction* (in addition to being correct by construction). These are based on two refined non-Boolean notions of winning strategies for rLTL games which both optimize the satisfaction of the specification.

The first notion, named *adaptive strategies*, uses automata-based runtime verification techniques [6] to monitor plays, detect bad moves of the environment, and adapt the actions of the system player to optimize the satisfaction of the winning condition. The second notion, named *strongly adaptive strategies*, is an extension of the first one that, in addition to being adaptive, also seeks to maximize the opportunity for the environment player to make bad moves. We show that both types of strategies can be computed using methods from automata theory and result in effective synthesis algorithms for reactive systems that are robust by construction and adapt to the environment at runtime.

After recapitulating rLTL in Section 2, we introduce adaptive strategies in Section 3 and show that one can compute such strategies in rLTL games in doubly-exponential time by reducing the problem to solving parity games [10]. In Section 4, we then turn to strongly adaptive strategies. It turns out that this type of strategy does not always exist, which we demonstrate through an example. Nevertheless, we give a doubly-exponential time algorithm that decides whether a strongly adaptive strategy exists, and, if this is the case, computes one. Our algorithm is based on reductions to a series of parity and obliging games [14]. As the LTL synthesis problem is 2EXPTIME-complete [29], which is a special case of the problems we consider here, computing both types of adaptive strategies is 2EXPTIME-complete as well. Furthermore, the size of the (strongly) adaptive strategies our algorithms compute is at most doubly exponential, matching the corresponding lower bound for LTL games, demonstrating that this bound is tight. Thus, our results show that adaptive robust-by-construction synthesis is asymptotically not harder than classical LTL synthesis.

All proofs omitted due to space restrictions can be found in the full version [24].

Related Work. Robustness in reactive synthesis has been addressed in various forms. A prominent example is work by Bloem et al. [7], which considers the synthesis of robust reactive systems from GR(1)-specifications. In subsequent work, Bloem et al. [9] have surveyed a large body of work on robustness in reactive synthesis and distilled three general categories: (i) "fulfill the guarantee as often as possible even if the environment assumption is violated", (ii) "if it is impossible to fulfill the guarantee, try to fulfill it whenever possible" and (iii) "help the environment to fulfill the assumption if possible". Prototypical examples include the work by Topcu et al. [37], Ehlers and Topcu [16], Chatterjee and Henzinger [12], Chatterjee et al. [14], and Bloem et al. [8].

However, our notion of adaptive strategies is more closely related to the notion of subgame perfect equilibrium [19]. A strategy profile is a subgame perfect equilibrium if it represents a Nash equilibrium of every subgame of the original game, i.e., the strategies are not only required to be optimal for the initial vertex but for every possible initial history of the game. Subgame perfect equilibria have been well studied in the context of graph games with LTL objectives [19,38]. Our results on adaptive strategies can be used to extend this concept to games with rLTL objectives. In particular, a pair of adaptive strategies for both players in such games forms a subgame perfect equilibrium and vice versa. Moreover, a subgame perfect equilibrium in such games is more fine-grained than subgame perfect equilibria in games with LTL objectives due to the many-valued semantics of rLTL. On the other hand, our notion of strongly adaptive strategies is more general than subgame perfect equilibria.

Also, the work of Almagor and Kupferman [1] is very similar to our notion of adaptive strategies. They introduced the notion of good-enough synthesis that is considered over a multi-valued semantics where the goal is to compute a strategy that achieves the highest possible satisfaction value. While some of the methods mentioned above do adapt to non-antagonistic behavior of the environment, we are not aware of any approach that would additionally optimize for the opportunities of the environment to act non-antagonistically, as our notion of strongly adaptive strategies does.

Quantitative objectives in graph-based games (and their combination with qualitative ones) have a rich history. Among the most prominent examples are mean-payoff parity games [13] and energy parity games [11]. The former type of game combines a parity winning condition (as the canonical representation for ω-regular properties) with a real-valued payout whose mean is to be maximized, while the latter type seeks to satisfy an ω-regular winning condition with the quantitative requirement that the level of energy during a play must remain positive. However, to the best of our knowledge, research in this field has focused on worst-case analyses with antagonistic environments.

Our notion of (strongly) adaptive strategies relies on central concepts introduced in the logic rLTL [36], a robust, many-valued extension of LTL. One of rLTL's key features is its syntactic similarity to LTL, which allows for a seamless and transparent transition from specifications expressed in LTL to specifications expressed in rLTL. Moreover, it is worth mentioning that rLTL has spawned numerous follow-up works, including rLTL model checking [2,3,4], rLTL runtime monitoring [21], and robust extensions of prompt LTL and Linear Dynamic Logic [27], as well as CTL [22].

Finally, let us highlight that preliminary results on adaptive strategies have been presented as a poster at the 24th ACM International Conference on Hybrid Systems: Computation and Control [23].

2 Preliminaries

In this section, we describe the syntax and semantics of Robust LTL and how it is different from classical LTL. Moreover, we discuss some important results on rLTL and introduce games with rLTL specifications.

Robust Linear Temporal Logic. We assume that the reader is familiar with Linear Temporal Logic [28]. We fix a finite non-empty set $\mathcal{P}$ of atomic propositions. The syntax of rLTL is similar to that of LTL with the only difference being the use of dotted temporal operators in order to distinguish them from LTL operators. More precisely, rLTL formulas are inductively defined as follows:

- each $p \in \mathcal{P}$ is an rLTL formula, and
- if φ and ψ are rLTL formulas, so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$, $\odot\varphi$ ("next"), $\boxdot\varphi$ ("always"), $\dot{\Diamond}\varphi$ ("eventually"), $\varphi\, \dot{\mathbf{R}}\, \psi$ ("release") and $\varphi\, \dot{\mathbf{U}}\, \psi$ ("until").

As already discussed, rLTL uses the set $\mathbb{B}_4 = \{1111, 0111, 0011, 0001, 0000\}$ of truth values, which are ordered as follows:

$$1111 > 0111 > 0011 > 0001 > 0000.$$

Intuitively, 1111 corresponds to "true", and the other four values correspond to different degrees of "false".

The rLTL semantics is a mapping $\mathcal{V}$, called *valuation*, that maps an infinite word $\alpha \in (2^{\mathcal{P}})^\omega$ and an rLTL formula φ to an element of $\mathbb{B}_4$. Before we define the semantics, we need to introduce some useful notation. Let $\alpha = \alpha_0\alpha_1 \cdots \in (2^{\mathcal{P}})^\omega$ be an infinite word. For $i \in \mathbb{N}$, let $\alpha_{i\ldots} = \alpha_i\alpha_{i+1}\cdots$ be the (infinite) suffix of α starting at position i. Also, for $1 \leq k \leq 4$, we let $V_k(\alpha, \varphi)$ denote the k-th entry of $\mathcal{V}(\alpha, \varphi)$, i.e., $\mathcal{V}(\alpha, \varphi) = V_1(\alpha, \varphi)V_2(\alpha, \varphi)V_3(\alpha, \varphi)V_4(\alpha, \varphi)$. Now, V is defined inductively as follows, where the semantics of Boolean connectives relies on *da*

Costa algebras [31]:

$$\mathcal{V}(\alpha, p) = \begin{cases} 0000 & \text{if } p \notin \alpha_0 \\ 1111 & \text{if } p \in \alpha_0 \end{cases} \qquad \mathcal{V}(\alpha, \neg\varphi) = \begin{cases} 0000 & \text{if } \mathcal{V}(\alpha,\varphi) = 1111 \\ 1111 & \text{otherwise} \end{cases}$$

$$\mathcal{V}(\alpha, \varphi \vee \psi) = \max\left\{\mathcal{V}(\alpha,\varphi), \mathcal{V}(\alpha,\psi)\right\} \quad \mathcal{V}(\alpha, \varphi \Rightarrow \psi) = \begin{cases} 1111 & \text{if } \mathcal{V}(\alpha,\varphi) \le \mathcal{V}(\alpha,\psi) \\ \mathcal{V}(\alpha,\psi) & \text{otherwise} \end{cases}$$

$$\mathcal{V}(\alpha, \varphi \wedge \psi) = \min\left\{\mathcal{V}(\alpha,\varphi), \mathcal{V}(\alpha,\psi)\right\} \qquad \mathcal{V}(\alpha, \odot\varphi) = \mathcal{V}(\alpha_{1\ldots}, \varphi)$$

$$\mathcal{V}(\alpha, \boxdot\varphi) = \left(\inf_{i\ge 0} V_1(\alpha_{i\ldots},\varphi), \sup_{j\ge 0}\inf_{i\ge j} V_2(\alpha_{i\ldots},\varphi), \inf_{j\ge 0}\sup_{i\ge j} V_3(\alpha_{i\ldots},\varphi), \sup_{i\ge 0} V_4(\alpha_{i\ldots},\varphi)\right)$$

$$\mathcal{V}(\alpha, \diamond\!\!\cdot\,\varphi) = \left(\sup_{i\ge 0} V_1(\alpha_{i\ldots},\varphi), \sup_{i\ge 0} V_2(\alpha_{i\ldots},\varphi), \sup_{i\ge 0} V_3(\alpha_{i\ldots},\varphi), \sup_{i\ge 0} V_4(\alpha_{i\ldots},\varphi)\right)$$

The semantics for the temporal operators $\mathbf{U}$ and $\mathbf{R}$ can be generalized similarly. We refer the reader to Tabuada and Neider [36] for more details.

Example 1. We can see that for the formula $\boxdot p$, the valuation $\mathcal{V}(\alpha, \boxdot p)$ can be expressed in terms of the LTL valuation function W by

$$\mathcal{V}(\alpha, \boxdot p) = W(\alpha, \Box p) W(\alpha, \Diamond\Box p) W(\alpha, \Box\Diamond p) W(\alpha, \Diamond p).$$

This evaluates to different values in $\mathbb{B}_4$ distinguishing various degrees of violations as seen in Section 1.

Example 2. Now let us see how the rLTL semantics for a specification of the form $\varphi \Rightarrow \psi$ captures robustness. Consider an instance where the environment assumption φ is $\boxdot p$ and the system guarantee ψ is $\boxdot q$ and assume the specification $\boxdot p \Rightarrow \boxdot q$ evaluates to 1111 for some infinite word. Let us see how the system behaves in response to various degrees of violation of the environment assumption.

– If p holds at all positions, then $\boxdot p$ evaluates to 1111. Hence, by the semantics of implication, $\boxdot q$ also evaluates to 1111, which means q holds at all positions. Therefore, the desired behavior of the system is retained when the environment assumption holds with no violation.
– If p holds eventually always but not always (a minor violation of $\boxdot p$), then $\boxdot p$ evaluates to 0111. Hence, $\boxdot q$ evaluates to 0111 or higher, meaning that q also needs to hold eventually always.
– Similarly, if p holds at infinitely (finitely) many positions, then q needs to hold at infinitely (finitely) many positions.

Hence, the semantics of $\boxdot p \Rightarrow \boxdot q$ captures the robustness property as desired. Furthermore, if $\boxdot p \Rightarrow \boxdot q$ evaluates to $b < 1111$, then $\boxdot p$ evaluates to a higher value than b, whereas $\boxdot q$ evaluates to b. So, the desired system guarantee is not satisfied. However, the value of $\boxdot p \Rightarrow \boxdot q$ still describes which weakened guarantee follows from the environment assumption.

From rLTL to Büchi Automata. Given an LTL formula φ, a generalized Büchi automaton (see [34] for a definition) with $O(2^{|\varphi|})$ states and $O(|\varphi|)$ accepting sets can be constructed that recognizes the infinite words satisfying φ [5], where $|\varphi|$ denotes the number of subformulas of φ. Using a similar method, Tabuada and Neider obtained the following result.

Theorem 1 ([36]). *Given an* rLTL *formula φ and a set of truth values $B \subseteq \mathbb{B}_4$, one can construct a generalized Büchi automaton $\mathcal{A}$ with $2^{O(|\varphi|)}$ states and $O(|\varphi|)$ accepting sets that recognizes the infinite words on which the value of φ belongs to B, i.e., $L(\mathcal{A}) = \{w \in (2^{\mathcal{P}})^\omega \mid \mathcal{V}(\alpha, \varphi) \in B\}$.*

rLTL Games. We consider infinite-duration two-player games over finite graphs with rLTL specifications. Here, we assume basic familiarity with games on graphs. Formally, an rLTL game $\mathcal{G} = (\mathcal{A}, \varphi)$ consists of (i) a finite, directed, labelled arena $\mathcal{A} = (V, E, \lambda)$ with $V = V_0 \uplus V_1$, an edge relation $E \subseteq V \times V$, and a labelling function $\lambda\colon V \to 2^{\mathcal{P}}$, and (ii) an rLTL formula φ over $\mathcal{P}$. The game is played by two players, Player 0 and Player 1, who construct a *play* $\rho = v_0 v_1 \cdots \in V^\omega$ by moving a token along the edges of the arena. A play $\rho = v_0 v_1 \cdots$ induces an infinite word $\lambda(\rho) = \lambda(v_0)\lambda(v_1)\cdots \in (2^{\mathcal{P}})^\omega$, and the *value* of the play, denoted by $\mathcal{V}(\rho)$, is the value of the formula φ on $\lambda(\rho)$. Player 0's objective is to maximize this value, while Player 1's objective is to minimize it.

Strategies. A *play prefix* is a finite, nonempty path $\mathtt{p} \in V^*$ in the arena. Then, a *strategy* for Player i, $i \in \{0, 1\}$, is a function $\sigma\colon V^* V_i \to V$ mapping each play prefix $\mathtt{p}$ ending in a vertex in V_i to one of its successors. Intuitively, a strategy prescribes Player i's next move depending on the play prefix constructed so far.

A strategy σ is *memoryless* if it only depends on the last vertex, i.e., for every prefix $\mathtt{p}$ ending in vertex v, it holds that $\sigma(\mathtt{p}) = \sigma(v)$. Moreover, we say a strategy has *memory size m* if there exists a finite state machine with output with m states computing the strategy (see Grädel et al. [18] for more details).

Next we define the plays that are consistent with a given strategy for Player i. Typically, this means that the token is placed at some initial vertex and then, whenever a vertex of Player i is reached, then Player i uses the move prescribed by the strategy for the current play prefix to extend this prefix. Note that the strategy does not have control over the initial placement of the token.

Here we will use a more general notion, inspired by previous work in optimal strategies for Muller games [17] and in subgame perfect equilibria in graph games [19,38]: the initial prefix over which the strategy does not have control over might be longer than just the initial vertex. This means strategies are also applicable to prefixes that where not constructed according to the strategy. However, crucially, the strategy still gets access to that prefix and therefore can base its decisions on the prefix it had no control over. This generality will turn out to be useful both when defining adaptive strategies and when combining strategies to obtain adaptive strategies.

Formally, for a play prefix $\mathtt{p} = v_0 v_1 \cdots v_n$ and a strategy σ for Player i, a play ρ is a $(\sigma, \mathtt{p})$-play if $\rho = \mathtt{p} v_{n+1} v_{n+2} \cdots$ with $v_{k+1} = \sigma(v_0 v_1 \cdots v_k)$ for all

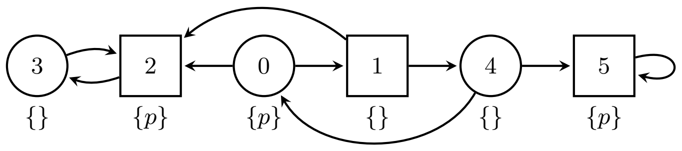

Fig. 1. First motivating example for adaptive strategies

$v_k \in V_i$ with $k \geq n$. Note that the prefix $\mathtt{p}$ is arbitrary here, i.e., it might not have been constructed following the strategy σ. Moreover, a $(\sigma, \mathtt{p})$-play prefix $\mathtt{pp}'$ is a prefix of a $(\sigma, \mathtt{p})$-play. We say that a play ρ starting in some vertex v is consistent with σ, if it is a (σ, v)-play (which is the classical notion of consistency). Finally, a play prefix $\mathtt{p}$ is consistent with σ if it is the prefix of some play that is consistent with σ.

In the paper introducing rLTL [36], Tabuada and Neider gave a doubly-exponential time algorithm that solves the classical rLTL synthesis problem, which is equivalent to solving the following problem.

Problem 1. Given an rLTL game $\mathcal{G}$, an initial vertex v_0 and a truth value $b \in \mathbb{B}_4$, compute a strategy σ (if one exists at all) for Player 0 such that every (σ, v_0)-play has value at least b.

Note that Tabuada and Neider were interested in strategies for Player 0 that enforce the value b from v_0, i.e., strategies such that every consistent play starting in the given initial vertex has at least value b. In contrast, we will compute strategies that are improvements in two dimensions: (i) they enforce the optimal value rather than a given one, and (ii) they do so from every possible play prefix, even if they did not have control over the prefix.

3 Adaptive Strategies

In this section, we start by presenting a motivating example, a game in which classical strategies for Player 0 are not necessarily optimal (in an intuitive sense). We then formalize this intuition by introducing adaptive strategies and give a doubly-exponential time algorithm to compute such strategies.

Motivating Example. Consider the arena given in Figure 1 (where Player 0's vertices are shown as circles and Player 1's vertices are shown as squares) with the rLTL specification $\varphi = \boxdot p$.

Suppose the token is initially placed at vertex 0. Considering Player 1 plays optimally, the token would eventually reach vertex 2, from which the best possible scenario for Player 0 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 0 is to try to visit vertex 2 infinitely often. That can be done by moving the token along one of the following edges every time the token reaches Player 0's vertices: $\{0 \to 1; 3 \to 2; 4 \to 0\}$. Note that the move $4 \to 0$ is irrelevant in this worst-case analysis, as vertex 4 is never reached if Player 1 plays optimally.

Suppose Player 1 makes a bad move by moving along $1 \to 4$. Then, Player 0 can force the play to eventually just stay at vertex 5, and hence, p holds almost always. However, the above classical strategy for Player 0 moves the play back to vertex 0, from which p might not hold almost always. Therefore, a better strategy for Player 0 is to move along $4 \to 5$ if the token reaches vertex 4 to get a play where p holds almost always; otherwise, enforce a play where p holds at infinitely many positions as earlier by moving along $0 \to 1$ and then $3 \to 2$ repeatedly.

In the worst case, i.e., if Player 1 does not make a bad move by reaching vertex 4, both strategies yield value 0011. However, if Player 1 does make a bad move by reaching vertex 4, the second strategy achieves value 0111 on some plays, while the second one does not. So, in the worst case analysis, both strategies are equally good, but if we assume that Player 1 is not necessarily antagonistic, then the second strategy is better as it is able to exploit the bad move by Player 1. We call such a strategy *adaptive* as it adapts its moves to achieve the best possible outcome after each bad move of the opponent. We will formalize this shortly.

To further illustrate the notion of adaptive strategies, consider another game with the arena shown in Figure 2 and with rLTL specification $\varphi' = (\odot \neg q \Rightarrow \boxdot p) \wedge (\odot q \Rightarrow \boxdot r)$. In this example, Player 1 has only two strategies starting from vertex 0: one moving the token along $0 \to 1$ and one moving the token along $0 \to 2$.

The best truth value Player 0 can enforce in this game is 0011. This is because Player 1 can move along the edge $0 \to 2$, which satisfies the second implication with value 1111 (as q does not occur), but also satisfies the premise of the first implication with value 1111. Hence, the value of the whole formula is the value of the subformula $\boxdot p$. The best value Player 0 can achieve for it is indeed 0011 by looping between vertices 3 and 2. His only other choice, i.e., to move to 4 eventually, only results in the value 0001.

However, if Player 1 does not take the edge $0 \to 2$ but instead moves to vertex 2 via vertex 1, Player 0 can gain from this *bad move* by instead moving to vertex 4. In that case, the formula is satisfied with truth value 1111. Thus, a strategy that adapts to the bad move by the opponent can achieve a better value than one that does not, if they do make a bad move.

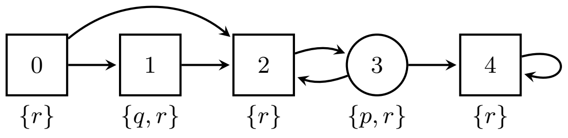

Fig. 2. Second motivating example for adaptive strategies

3.1 Definitions

Recall that a $(\sigma, \mathtt{p})$-play for a strategy σ for Player i and a play prefix $\mathtt{p}$ (not necessarily consistent with σ) is an extension of $\mathtt{p}$ by σ, i.e., Player i uses the strategy σ to extend the play prefix p they had not control over, while still taking the prefix $\mathtt{p}$ into account when making the decisions. We say that a strategy σ for Player 0 *enforces* a truth value of b from a play prefix $\mathtt{p}$, if we have $\mathcal{V}(\rho) \geq b$ for every $(\sigma, \mathtt{p})$-play ρ. Similarly, we say a strategy τ for Player 1 enforces a truth value of b from a play prefix $\mathtt{p}$, if we have $\mathcal{V}(\rho) \leq b$ for every $(\tau, \mathtt{p})$-play ρ. This conforms to our intuition that Player 0 tries to maximize the truth value while Player 1 tries to minimize it. Moreover, we say Player i can enforce a value b from some prefix $\mathtt{p}$ if they have a strategy that enforces b from $\mathtt{p}$.

Remark 1. Let $\mathtt{p}$ be a play prefix. If Player 0 can enforce value b_0 from $\mathtt{p}$ and Player 1 can enforce value b_1 from $\mathtt{p}$ then $b_0 \leq b_1$.

For example, consider the game given in Figure 1 with the rLTL specification $\boxdot p$. Using the analysis given in Section 3, we can see that Player 0 can enforce 0111 and 0011 from prefixes 014 and 012, respectively, by moving the token along $\{0 \to 1; 4 \to 5; 3 \to 2\}$. It is easy to check that these are the best values Player 0 can enforce from those prefixes as Player 1 can enforce the same values from these prefixes.

We are interested in a strategy that enforces the best possible value from each play prefix. This is formalized as follows.

Definition 1 (Adaptive Strategies). *In an* rLTL *game, a strategy σ_0 for Player* 0 *is adaptive if from every play prefix* $\mathtt{p}$*, no strategy for Player* 0 *enforces a better truth value than σ_0, that is, if some strategy σ for Player* 0 *enforces a truth value of b from* $\mathtt{p}$*, then σ_0 also enforces the value b from* $\mathtt{p}$*.*

Note that $\mathtt{p}$ is not required to be consistent with σ_0 in the above definition, i.e., an adaptive strategy achieves the best possible outcome from every possible play prefix (even for those it had no control over when they are constructed). Also, let us mention that a dual notion can be defined for Player 1.

As mentioned earlier, one can notice that the concept of adaptive strategies is similar to the concept of subgame perfect equilibrium [19]. Furthermore, adaptive strategies can be used to extend the notion of subgame perfect equilibrium to rLTL games as in the following remark.

Remark 2. Given an rLTL game, a pair of adaptive strategies for both players forms a subgame perfect equilibrium for the game, and vice versa.

Here, we prefer the notion of adaptive strategies over the notion of subgame perfect equilibria, as we focus on strategies for Player 0 and generally disregard strategies of Player 1.

3.2 Computing Adaptive Strategies

Now, to synthesize an adaptive strategy, we need to monitor the bad moves of the opponent at runtime by keeping track of the best value that can be enforced from the current play prefix. To do that, using the idea of automata-based runtime verification [6], we construct multiple parity automata to monitor the bad moves of the opponent and then we synthesize adaptive strategies by using a reduction to parity games (see [18] for definitions).

Given an rLTL game $\mathcal{G} = (\mathcal{A}, \varphi)$ with $\mathcal{A} = (V, E, \lambda)$, we proceed as follows:

1. We construct generalized (non-deterministic) Büchi automata $\mathcal{A}^b$ such that $L(\mathcal{A}^b) = \{w \in (2^{\mathcal{P}})^\omega \mid \mathcal{V}(w, \varphi) \geq b\}$ for all $b \in \mathbb{B}_4$.
2. We determinize each $\mathcal{A}^b$ to obtain a deterministic parity automaton $\mathcal{C}^b$ with the same language.
3. For each b, we construct a parity game $\mathcal{G}^b$ by taking the product of the arena $\mathcal{A}$ and the parity automaton $\mathcal{C}^b$.
4. We solve the above parity games $\mathcal{G}^b$ [10], yielding, for each truth value b, a finite-state winning strategy for the original game $\mathcal{G}$ with value b (if one exists).
5. We combine all these winning strategies for Player 0 computed in the last step to obtain an adaptive strategy σ for Player 0.

Let us now explain each step in more detail.

Step 1. We construct the generalized non-deterministic Büchi automata $\mathcal{A}^b$ such that $L(\mathcal{A}^b) = \{w \in (2^{\mathcal{P}})^\omega \mid \mathcal{V}(w, \varphi) \geq b\}$ for all $b \in \mathbb{B}_4$. By Theorem 1, the automaton $\mathcal{A}^b$ has $2^{O(|\varphi|)}$ states and $O(|\varphi|)$ accepting sets.

Step 2. We determinize each $\mathcal{A}^b$ to get a deterministic parity automaton $\mathcal{C}^b = (Q^b, 2^{\mathcal{P}}, q_0^b, \delta^b, \Omega^b)$ with $2^{2^{O(|\varphi|)}}$ states and $2^{O(|\varphi|)}$ colors [34].

Step 3. We construct the (unlabelled) product arena $\mathcal{A}^b = (V^b, E^b)$ of the arena $\mathcal{A} = (V, E, \lambda)$ and the parity automaton $\mathcal{C}^b$ such that $V^b = V \times Q^b$, $V_i^b = V_i \times Q^b$ for $i \in \{0, 1\}$, and

$$((v, q), (v', q')) \in E^b \text{ if and only if } (v, v') \in E \text{ and } \delta^b(q, \lambda(v)) = q'.$$

The function $\bar{\Omega}^b$ assigns colors to the vertices such that $\bar{\Omega}^b(v, q) = \Omega^b(q)$. The desired parity games are the $\mathcal{G}^b = (\mathcal{A}^b, \bar{\Omega}^b)$ with $b \in \mathbb{B}_4$.

It is easy to verify that Player 0 wins a play $\rho' = (v_0, q_0^b)(v_1, q_1^b) \cdots$ in $\mathcal{G}^b$ if and only if the value of the play $\rho = v_0 v_1 \cdots$ in $\mathcal{G}$ is at least b. Furthermore, given a path $\rho = v_0 v_1 \cdots v_k$ in $\mathcal{A}$, there is a unique path of the form $\rho' = (v_0, q_0^b)(v_1, q_1^b) \cdots (v_k, q_k^b)$ in $\mathcal{A}^b$, that is when $q_{i+1}^b = \delta^b(q_i^b, v_i)$ for all $0 \leq i \leq k-1$.

Since winning a play in $\mathcal{G}^b$ is equivalent to the corresponding play in $\mathcal{G}$ satisfying φ with truth value b or greater, we can characterize the enforcement of b in $\mathcal{G}$ by the winning region of Player 0 in $\mathcal{G}^b$, i.e., the set of vertices from which Player 0 has a winning strategy. This can easily be shown by simulating a winning strategy from (v, q) to extend the play prefix $\mathtt{p}$ and vice versa.

Remark 3. Fix a play prefix $\mathtt{p}$ in the rLTL game $\mathcal{G}$, and let (v, q^b) be the last vertex of the corresponding play in the parity game $\mathcal{G}^b$ for some b. Then, Player 0 can enforce b from $\mathtt{p}$ if and only if (v, q^b) is in his winning region of $\mathcal{G}^b$.

Step 4. We solve the resulting parity games $\mathcal{G}^b$ and determine the winning regions $\mathrm{Win}(\mathcal{G}^b)$ of Player 0 and uniform memoryless winning strategies σ^b for Player 0 that are winning from every vertex in the corresponding winning region. The parity games have $n = |V| \cdot 2^{2^{O(|\varphi|)}}$ vertices and $k = 2^{O(|\varphi|)}$ colors. Since $k < \lg(n)$, these can be solved in time $O(n^5) = |V|^5 \cdot 2^{2^{O(|\varphi|)}}$ [10].

Step 5. Consider the extended rLTL game $\mathcal{G}' = (\mathcal{A}', \varphi)$, where $\mathcal{A}' = (V', E', \lambda')$ with $V' = V \times Q^{0000} \times \cdots \times Q^{1111}$,

$$E' = \big\{ \big((v_1, q_1^{0000}, \ldots, q_1^{1111}), (v_2, q_2^{0000}, \ldots, q_2^{1111})\big) \mid \\ (v_1, v_2) \in E \text{ and } \delta^b(q_1^b, \lambda(v)) = q_2^b \text{ for all } b \in \mathbb{B}_4 \big\},$$

and λ' such that $\lambda'(v, q^{0000}, \ldots, q^{1111}) = \lambda(v)$ for all $v \in V$ and $q^b \in Q^b$.

It is easy to see that there is a one to one correspondence between the plays in both games $\mathcal{G}$ and $\mathcal{G}'$. Besides that, the rLTL specification is also the same in both games. Therefore, computing an adaptive strategy in the game $\mathcal{G}$ is equivalent to computing one in the game $\mathcal{G}'$. Now using the analysis given in Step 3, we have the following in the rLTL game $\mathcal{G}'$:

- A vertex v' is in $\mathrm{E}_{\geq b} = \{(v, q^{0000}, \ldots, q^{1111}) \in V' \mid (v, q^b) \in \mathrm{Win}(\mathcal{G}^b)\}$ if and only if Player 0 can enforce b from every play prefix in $\mathcal{G}'$ ending in v'.
- Using these sets, we now define the set $\mathrm{E}_{=b}$ of vertices from which the maximum value Player 0 can enforce is b. Formally, this set is given by

$$\mathrm{E}_{=b} = \begin{cases} \mathrm{E}_{\geq 1111} & \text{if } b = 1111, \\ \mathrm{E}_{\geq b} \setminus \mathrm{E}_{\geq b+1} & \text{if } b < 1111, \end{cases}$$

  where $b+1$ is the smallest value bigger than $b < 1111$. Note that the sets $\mathrm{E}_{=b}$ form a partition of the vertex set of $\mathcal{G}'$.

Furthermore, it is easy to see that if a play ρ satisfies a parity objective then every play sharing a suffix with ρ also satisfies the parity objective. Since the game $\mathcal{G}'$ is a product of parity games and since we have characterized the enforcement of truth values via the membership in the winning regions of the parity games (see Remark 3), the next remark follows.

Remark 4. In the rLTL game $\mathcal{G}'$, for two play prefixes $\mathtt{p}_1, \mathtt{p}_2$ ending in the same vertex, the following holds: if a memoryless strategy σ for Player 0 enforces a truth value b from $\mathtt{p}_1$, then it also enforces the value b from $\mathtt{p}_2$.

Then, we can see that if the token stays in $\mathrm{E}_{=b}$ for some b, then Player 0 can simulate the strategy σ^b for $\mathcal{G}^b$ to enforce the value b in $\mathcal{G}'$. Therefore, we obtain a memoryless adaptive strategy σ for Player 0 in the game $\mathcal{G}'$ as follows: for every vertex $(v, q^{0000}, \ldots, q^{1111})$ in $\mathrm{E}_{=b}$, we define $\sigma(v, q^{0000}, \ldots, q^{1111})$ to be the unique successor of $(v, q^{0000}, \ldots, q^{1111})$ in $\mathcal{G}'$ that corresponds to the successor $\sigma^b(v, q^b)$ of (v, q^b) in $\mathcal{G}^b$. Thus, σ simulates the strategy σ^b for the largest b such that the value b can be enforced (which is exactly what σ^b does from such a prefix). Hence, it is an adaptive strategy for Player 0 in $\mathcal{G}'$.

Finally, using the strategy σ, one can compute a corresponding strategy in the game $\mathcal{G}$ with memory $Q^{0000} \times Q^{0001} \times \cdots \times Q^{1111}$, which is used to simulate the positional strategy σ. The resulting finite-state strategy is an adaptive strategy for Player 0 in $\mathcal{G}$.

Note that the adaptive strategy in $\mathcal{G}$ is of doubly-exponential size in $|V|$ and $|\varphi|$. This upper bound is tight, since there is a doubly-exponential lower bound on the size of winning strategies strategies for LTL games [30], which can be lifted to rLTL games.

Similarly, one can compute an adaptive strategy for Player 1. Hence, an adaptive strategy for both players in an rLTL game can be computed in time doubly-exponential in the size of the formula.

Theorem 2. *Given an* rLTL *game, an adaptive strategy of a player can be computed in doubly-exponential time. Moreover, each player has an adaptive strategy with doubly-exponential memory size.*

Note that adaptive strategies enforce the best possible value from the given prefix. This value can be obtained at runtime as follows: Given a play prefix `p` ending in some vertex v, let $(q^{0000}, \ldots, q^{1111})$ be the state of the automaton implementing the adaptive finite-state strategy computed above is in after the prefix `p`. Note that this state has to be tracked to determine the next move the strategy prescribes at prefix `p` (in case $v \in V_0$). Then, there is a unique b such that $(v, q^{0000}, \ldots, q^{1111}) \in \mathrm{E}_{=b}$. Then, the value currently enforced by the adaptive strategy is b, which, by construction, is the maximal one that can be enforced from `p`.

4 Strongly Adaptive Strategies

In the previous section, we have argued the importance of adaptive strategies and proved that in every rLTL game both players have an adaptive strategy. Intuitively, such a strategy exploits bad moves of the opponent to always enforce the best truth value possible after a given prefix. However, such a strategy does not necessarily seek out opportunities for the opponent to make bad moves. We argue that this property implies that some adaptive strategies are more desirable than others, which leads us to the notion of strongly adaptive strategies.

In this section, we define strongly adaptive strategies, which are based on a fine-grained analysis of the possibilities a strategy gives the opponent to make bad moves and the resulting outcomes of such bad moves. We show that strongly adaptive strategies do not exist in every rLTL game. This is in stark contrast to adaptive strategies, which always exists. Nevertheless, we give a doubly-exponential time algorithm that decides whether a strongly adaptive strategy exists and, if yes, computes one.

4.1 Bad Moves

We already have used the notion of bad moves in Section 3 in an intuitive, but informal, way. Formally, we say a play $\rho = v_0 v_1 \cdots$ contains a bad move of Player i at position $j > 0$ if the player can enforce some value b from the prefix $v_0 \cdots v_{j-1}$ but can no longer enforce the value b from the prefix $v_0 \cdots v_j$. Note that the position j is the target of the bad move. Moreover, note that moving from $v_0 \cdots v_{j-1}$ to v_j can only be a bad move for Player i if it is Player i's turn at v_{j-1}. Also, there must be some other edge from v_{j-1} to a vertex $v \neq v_j$ so that they can still enforce b from $v_0 \cdots v_{j-1} v$.

For the example given in Figure 1, we know that Player 1 can enforce the value 0011 from 01 (by moving the token from 01 to 2). Suppose she moves the token from 01 to 4 instead. Then, Player 0 can enforce 0111 by visiting vertex 5. Hence Player 1 can no longer enforce 0011 from 014. Therefore, the move from prefix 01 to vertex 4 made by Player 1 is bad.

Note that if Player 1 makes a bad move from a play prefix $\mathtt{p}$ to vertex v, then the maximum value Player 0 can enforce from $\mathtt{p}v$ is strictly larger than the maximum value he can enforce from $\mathtt{p}$. Hence, if the maximum value Player 0 can enforce from a play prefix is 1111, then Player 1 can not make any bad move from that prefix. Moreover, assuming Player 0 does not make any bad move, the maximum value Player 0 can enforce from any play prefix can increase at most four times during a play. Thus, Player 1 can make at most four bad moves against an adaptive strategy, because such a strategy does not make any bad moves.

Remark 5. Let σ be an adaptive strategy and $\mathtt{p}$ a play prefix (not necessarily consistent with σ). Then, every $(\sigma, \mathtt{p})$-play $\rho = v_0 v_1 \cdots$ contains at most four bad moves of Player 1 after $\mathtt{p}$. Also, if there is no bad move by Player 1 at positions $j_0, j_0 + 1, \ldots, j_1$ in ρ, then σ enforces the same truth values from every prefix of the form $v_0 \cdots v_j$ with $j_0 - 1 \leq j < j_1$.

Our next example shows that an adaptive strategy does not actively seek out opportunities for the opponent to make bad moves, it just exploits those made.

4.2 Motivating Example

Recall the example given in Figure 1. The strategy for Player 0 given by $\{0 \to 1; 3 \to 2; 4 \to 5\}$ is adaptive: if Player 1 makes a bad move by moving from 1 to 4, then moving from 4 to 5 improves the value of the play to 0111. Such an improvement can only be enforced after the bad move.

Another adaptive strategy for Player 0 is to move along $0 \to 2$ directly in his first move and then move along $3 \to 2$ every time. Then, the token can never reach vertex 1. Hence, Player 1 can never make a bad move. However, it also means that there can not be a play with value 0111. By contrast, if Player 0 moves along $0 \to 1$, there is a chance of getting such plays (when Player 1 makes a bad move of $1 \to 4$). Therefore, using the earlier strategy of moving the token along $0 \rightarrow 1$, Player 0 might be able to enforce 0111 at some point, but he can never achieve the value 0111 when moving directly to vertex 2.

Similarly, in many games, a player may have two (or more) optimal choices to move the token from some prefix. In such situations, that player should compare the bad moves their opponent can make in both choices and determine the choice in which they can enforce the best value after a bad move has been made by the opponent. To capture this, we refine the notion of adaptive strategies by introducing strongly adaptive strategies, which are, in a sense to be formalized below, the best adaptive strategies.

4.3 Definitions

In this section, we introduce the necessary machinery to define strongly adaptive strategies for Player 0. Throughout this section, we are concerned with ranking adaptive strategies according to the number of bad moves they allow Player 1 to make, and on the effect these moves have. For the sake of conciseness, unless stated otherwise, from now on a bad move always refers to a bad move by Player 1.

We begin by introducing a ranking of plays and then lift this to strategies. As the number of bad moves in one play is bounded by four, this results in at most five truth values that can be enforced from prefixes of the play, i.e., the one that is enforced before the first bad move, and the ones after each bad move. If Player 1 makes less than four bad moves during a play, we use the symbol $\perp \notin \mathbb{B}_4$ to signify this.

We collect this information in a summary, a five-tuple $(b_0, \dots, b_k, \perp, \dots, \perp) \in (\mathbb{B}_4 \cup \{\perp\})^5$ such that $\perp \neq b_0 < b_1 < \cdots < b_k$. The set of all summaries is denoted by $\mathcal{S}$.

Fix an adaptive strategy σ for Player 0, a play prefix $\mathtt{p}$ not necessarily consistent with σ, and a $(\sigma, \mathtt{p})$-play ρ, and let $0 \leq k \leq 4$ be the number of bad moves by Player 1 after $\mathtt{p}$. Define $\mathtt{p}_0 = \mathtt{p}$ and let $\mathtt{p}_j$, for $1 \leq j \leq k$, be the prefix of ρ ending at the position of the j-th bad move. Due to Remark 5, these prefixes contain information about all possible truth vales that are enforced by Player 0 from prefixes of ρ. We employ summaries to capture the values a given *strategy* σ enforces from these prefixes. Formally, for $0 \leq j \leq k$, let b_j be the maximal value that σ enforces from $\mathtt{p}_j$. As σ is adaptive, these values are strictly increasing. So, we can define the summary $\mathrm{smry}(\sigma, \mathtt{p}, \rho) = (b_0, \dots, b_k, \perp, \dots, \perp)$. Intuitively, the summary collects all information about which truth values the strategy σ enforces after each bad move has been made. If there are less than four bad moves in ρ after $\mathtt{p}$, then we fill the summary with $\perp$'s to obtain a vector of length five.

We will use such summaries to compare strategies. To do so, we compare summaries in lexicographic order $\leq_{\mathrm{lex}}$ with $\perp$ being the smallest element. In other words, we prefer larger truth values of smaller ones and prefer the opportunity for a bad move over the impossibility of a bad move.

Example 3. Consider again the game in Figure 1. Let σ_1 be the memoryless Player 0 strategy always making the moves $\{0 \to 1; 3 \to 2; 4 \to 5\}$. Then,

$$\mathrm{smry}(\sigma_1, 0, 0145^\omega) = \mathrm{smry}(\sigma_1, 01, 0145^\omega) = (0011, 0111, \bot, \bot, \bot)$$

and $\mathrm{smry}(\sigma_1, 014, 0145^\omega) = (0111, \bot, \bot, \bot, \bot)$ because the play 0145^ω does not contain a bad move of Player 1 after 014. In addition, $\mathrm{smry}(\sigma_1, \mathtt{p}, 01(23)^\omega) = (0011, \bot, \bot, \bot, \bot)$ for every prefix $\mathtt{p}$ of $01(23)^\omega$, as the play does not contain any bad move of Player 1.

Let σ_2 now be the memoryless Player 0 strategy given by $\{0 \to 2; 3 \to 2; 4 \to 5\}$. Then, we have $\mathrm{smry}(\sigma_2, \mathtt{p}, 0(23)^\omega) = (0011, \bot, \bot, \bot, \bot)$ for every prefix $\mathtt{p}$ of $0(23)^\omega$ because the play does not contain bad moves of Player 1.

We continue by listing some simple properties of summaries that are useful later on. Consider the prefixes 0, 01, 014 of 0145^ω in Example 3. The former two have the same summary s, while the summary of the latter is obtained by shifting s to the left. Note that moving from 01 to 4 is a bad move of Player 1, while moving from 0 to 1 is not. By inspecting the definition of play summaries, it is clear that extending plays by bad moves corresponds to a left shift, while Remark 5 implies that the absence of bad moves keeps summaries stable.

To formalize this, we use the following notation: for $s = (b_0, \dots, b_k, \bot, \dots, \bot) \in \mathcal{S}$ with $k > 0$ let $\mathrm{lft}(s) = (b_1, \dots, b_k, \bot, \dots, \bot) \in \mathcal{S}$, i.e., we shift s to the left and fill the last entry with a $\bot$. As entries in summaries are strictly increasing, we have $\mathrm{lft}(s) >_{\mathrm{lex}} s$ for every s with at least two non-$\bot$ entries.

Remark 6. Let σ be an adaptive strategy for Player 0, let $\mathtt{p}$ be a play prefix, and let $\rho = v_0 v_1 \cdots$ be a $(\sigma, \mathtt{p})$-play. Further, let $n = |\mathtt{p}|$, i.e., v_{n-1} is the last vertex of $\mathtt{p}$, and note that ρ is also a $(\sigma, \mathtt{p}v_n)$-play.

If ρ has a bad move at position n, then $\mathrm{smry}(\sigma, \mathtt{p}v_n, \rho) = \mathrm{lft}(\mathrm{smry}(\sigma, \mathtt{p}, \rho))$ (reflecting the fact that ρ has one bad move less after $\mathtt{p}v_n$ than after $\mathtt{p}$), otherwise we have $\mathrm{smry}(\sigma, \mathtt{p}v_n, \rho) = \mathrm{smry}(\sigma, \mathtt{p}, \rho)$. Note that we have kept σ and ρ fixed and just added a vertex to the prefix we consider.

As seen above, a bad move shifts the summary to the left. The following remark shows a dual result, allowing us to determine the summary of a play prefix of length one from the summary of play prefix up to the first bad move. In Example 3, note that the strategy σ_1 (using the edges $\{0 \to 1; 3 \to 2; 4 \to 5\}$) enforces value 0011 from 0, i.e., the first entry of $\mathrm{smry}(\sigma_1, 0, 0145^\omega)$ is 0011. The play 0145^ω has its first bad move of Player 1 at position 2, and the corresponding summary is $\mathrm{smry}(\sigma_1, 014, 0145^\omega) = (0111, \bot, \bot, \bot, \bot)$. Hence, $\mathrm{smry}(\sigma_1, 0, 0145^\omega)$ must be the "concatenation" $(0011, 0111, \bot, \bot, \bot)$ of 0011 and $(0111, \bot, \bot, \bot, \bot)$ (with the last $\bot$ removed). In general, we have the following property.

Remark 7. Let $s = (b_0, \dots, b_k, \bot, \dots, \bot) \in \mathcal{S}$ with $k > 0$ and let v be a vertex. Let σ be an adaptive strategy such that b_0 is the maximal value that σ enforces from v and let ρ be a (σ, v)-play with at least one bad move, and let $\mathtt{p}$ be the prefix of ρ ending at the position of the first bad move. Then, $\mathrm{smry}(\sigma, \mathtt{p}, \rho) = \mathrm{lft}(s)$ if and only if $\mathrm{smry}(\sigma, v, \rho) = s$.

Again, recall Example 3, and consider the plays $\rho_b = 0145^\omega$ (with a bad move by Player 1) and $\rho_n = 01(23)^\omega$ (without a bad move), which are both $(\sigma_1, 0)$-plays. We have $\mathrm{smry}(\sigma_1, 0, \rho_b) = (0011, 0111, \bot, \bot, \bot)$ and $\mathrm{smry}(\sigma_1, 0, \rho_n) = (0011, \bot, \bot, \bot, \bot)$. Disregarding the $\bot$'s the summary of ρ_n can be seen as a strict prefix of the summary of ρ_b. Note that $(0011, \bot, \bot, \bot, \bot) <_{\mathrm{lex}} (0011, 0111, \bot, \bot, \bot)$.

In general, fix a strategy σ, a play prefix $\mathtt{p}$, and a $(\sigma, \mathtt{p})$-play ρ with $\mathrm{smry}(\sigma, \mathtt{p}, \rho) = (b_0, \dots, b_k, \bot, \dots, \bot)$. Then, for every $k' < k$ there is a $(\sigma, \mathtt{p})$-play ρ' with $\mathrm{smry}(\sigma, \mathtt{p}, \rho') = (b_0, \dots, b_{k'}, \bot, \dots, \bot)$, i.e., any play where Player 1 stops making bad moves after the first k' ones (recall that making bad moves is a choice).

To formalize this, we say that a summary $(b_0, \dots, b_k, \bot, \dots, \bot)$ is a strict prefix of a summary $(b'_0, \dots, b'_{k'}, \bot, \dots, \bot)$ if $k < k'$ and $b_j = b'_j$ for all $0 \le j \le k$, i.e., we only consider non-$\bot$ entries. Now, fix $(\sigma, \mathtt{p})$-plays ρ, ρ'. We say that ρ is $(\sigma, \mathtt{p})$-covered by ρ' if $\mathrm{smry}(\sigma, \mathtt{p}, \rho)$ is a strict prefix of $\mathrm{smry}(\sigma, \mathtt{p}, \rho')$. Also, we say that ρ is a $(\sigma, \mathtt{p})$-uncovered play if there is no $(\sigma, \mathtt{p})$-play ρ' that covers it. When σ and $\mathtt{p}$ are clear from context, we drop them and say that a play is uncovered. In the example, ρ_n is $(\sigma_1, 0)$-covered by ρ_b, which is $(\sigma_1, 0)$-uncovered.

Now, we lift summaries from plays to strategies by defining $\mathrm{smry}(\sigma, \mathtt{p})$ as the lexicographical minimum over all $\mathrm{smry}(\sigma, \mathtt{p}, \rho)$ where ρ ranges over $(\sigma, \mathtt{p})$-uncovered plays. Note that if ρ $(\sigma, \mathtt{p})$-covers ρ', then the summary of ρ is a strict prefix of the summary of ρ' and, therefore, strictly smaller. Our definition of $\mathrm{smry}(\sigma, \mathtt{p})$ discards such plays when computing the minimum, but the information is not lost as it appears as a prefix of a covering play.

In the running example, we have $\mathrm{smry}(\sigma_1, 0) = (0011, 0111, \bot, \bot, \bot)$ and $\mathrm{smry}(\sigma_2, 0) = (0011, \bot, \bot, \bot, \bot)$.

Remark 8. Let σ be an adaptive strategy for Player 0 and let $\mathtt{p}$ be a play prefix. If $\mathrm{smry}(\sigma, \mathtt{p}) = s$ for some $s \in \mathcal{S}$, then there exists a $(\sigma, \mathtt{p})$-uncovered play ρ such that $\mathrm{smry}(\sigma, \mathtt{p}, \rho) = s$.

Finally, we are ready to formalize our intuitive notion of strongly adaptive strategies, i.e., adaptive strategies that seek out opportunities for the opponent to make bad moves. Recall that summaries record the possibility, and the effect, of Player 1 making bad moves. So, we intuitively say a strategy is strongly adaptive if it maximizes the summaries globally.

Recall that a strategy is adaptive if the value it enforces from any possible play prefix is as large as the value any other strategy enforces from that prefix. Analogously, a strategy is strongly adaptive if its summary for every play prefix is as good as the summary from the play prefix for any other strategy.

Definition 2. *An adaptive strategy σ_0 is strongly adaptive if* $\mathrm{smry}(\sigma_0, \mathtt{p}) \ge_{\mathrm{lex}} \mathrm{smry}(\sigma, \mathtt{p})$ *for every adaptive strategy σ and every play prefix* $\mathtt{p}$.

Before we start our proof, let us introduce one more useful bit of notation. For every play prefix $\mathtt{p}$, let $\mathrm{smry}(\mathtt{p})$ denote the lexicographical maximum of $\mathrm{smry}(\sigma, \mathtt{p})$ over all adaptive strategies σ for Player 0 in the game $\mathcal{G}$, i.e.,

$$\mathrm{smry}(\mathtt{p}) = \max_{\sigma} \mathrm{smry}(\sigma, \mathtt{p}),$$

where σ ranges over all adaptive strategies for Player 0.

Note that every strongly adaptive strategy is adaptive by definition, and the first entry of smry($\mathtt{p}$) is equal to the maximal value that can be enforced from $\mathtt{p}$. However, as argued above, not every adaptive strategy is strongly adaptive, so in particular a strongly adaptive strategy for Player 0 never makes a bad move (of Player 0).

4.4 Existence of Strongly Adaptive Strategies

While strongly adaptive strategies generalize adaptive strategies, there is a catch in the definition: The former may not always exist, whereas the latter always do. To show this, consider the graph given in Figure 3 with initial vertex 0 and the formula $\varphi = \boxdot p$. It is clear that Player 0 can enforce 0011 from every play prefix in $0(10)^*$ by eventually moving to vertex 3. And if at some point, Player 1 makes the bad move $1 \to 2$, then Player 0 enforces 0111 as the token stays at vertex 2 forever. However, every adaptive strategy for Player 0 has to eventually visit vertex 3, unless Player 1 makes a bad move prior.

Note that Player 1 can only make a bad move at vertex 1, so visiting 1 once more when at vertex 0 instead of moving to vertex 3 gives her another chance to make a bad move. So, to optimize the enforced value under one bad move, Player 1 should stay in the loop between 0 and 1 forever. However, this is not the optimal behavior if no bad move occurs, as looping yields a value of 0000, which is smaller than the value 0011 that is achieved by eventually moving to 3.

Formally, for $n \geq 0$, let σ_n be the strategy such that $\sigma_n(0(10)^{n'}) = 1$ for all $n' < n$ and $\sigma_n(0(10)^{n'}) = 3$ for all $n' \geq n$, i.e., σ_n gives Player 1 n chances to make a bad move and then moves to 3, thereby preventing her from making a bad move. Note that each of the σ_n is adaptive, but σ_{n+1} gives Player 1 more opportunities to make a bad move than σ_n, namely for the prefix $0(10)^n$.

Fix some n. There are only two $(\sigma_{n+1}, 0(10)^n)$-plays, i.e., $0(10)^n10(34)^\omega$ (Player 1 does not make a bad move) and $0(10)^n12^\omega$ (Player 1 makes a bad move). Then, $\mathrm{smry}(\sigma_{n+1}, 0(10)^n, 0(10)^n10(34)^\omega) = (0011, \perp, \perp, \perp, \perp)$ as well as $\mathrm{smry}(\sigma_{n+1}, 0(10)^n, 0(10)^n12^\omega) = (0011, 0111, \perp, \perp, \perp)$. Hence, we conclude $\mathrm{smry}(\sigma_{n+1}, 0(10)^n) = (0011, 0111, \perp, \perp, \perp)$, as the former is covered by the latter.

Towards a contradiction assume there is a strongly adaptive strategy σ. By definition, we have

$$\mathrm{smry}(\sigma, 0(10)^n) \geq_{\mathrm{lex}} \mathrm{smry}(\sigma_{n+1}, 0(10)^n) = (0011, 0111, \perp, \perp, \perp) \qquad (1)$$

for every n. As we have $\mathrm{smry}(\sigma, 0(10)^n) <_{\mathrm{lex}} (1111, \perp, \perp, \perp, \perp)$ (p does not hold at vertex 0), σ must give Player 1 the chance to make at least one bad move after the prefix $0(10)^n$. So, we must have $\sigma(0(10)^n) = 1$, as Player 1 can only make a bad move at vertex 1.

Thus, the play $(01)^\omega$ (with value 0000) is a $(\sigma, 0(10)^n)$-play for every n, i.e., σ only enforces 0000 from every such prefix. Hence, the first entry of $\mathrm{smry}(\sigma, 0(10)^n)$ is 0000 for every n. This contradicts Inequality (1). Therefore, σ is not strongly adaptive, i.e., Player 0 does not have a strongly adaptive strategy in the game.

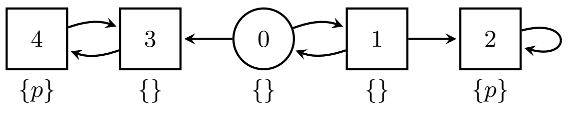

Fig. 3. An rLTL game with no strongly adaptive strategy

As strongly adaptive strategies do not necessarily exist, we are interested in the following problem.

Problem 2. Given an rLTL game, determine whether a strongly adaptive strategy for Player 0 exists and, if yes, compute one.

4.5 Computing Strongly Adaptive Strategies

We solve Problem 2 for an rLTL game $\mathcal{G} = (\mathcal{A}, \varphi)$ by constructing the parity games $\mathcal{G}^b$ for each b and the extended game $\mathcal{G}' = (\mathcal{A}', \varphi)$ as in the algorithm given in Section 3.2. Recall that Player 0 wins $\mathcal{G}^b$ if and only if he can enforce b in $\mathcal{G}$ and that $\mathcal{G}'$ is the product of the $\mathcal{G}^b$. As we have described in Step 5 of that algorithm, it is easy to see that solving Problem 2 for the game $\mathcal{G}$ is equivalent to solving the problem for game $\mathcal{G}'$. Hence, from now on, we only consider $\mathcal{G}'$ and show properties for the game $\mathcal{G}'$, which we can use later to compute a strongly adaptive strategy in $\mathcal{G}$. This strategy can then be transformed into a strongly adaptive strategy for $\mathcal{G}$. In the following, it is often useful to focus on one truth value by equipping $\mathcal{G}'$ with the parity condition of $\mathcal{C}_b$ for some b: a vertex $(v, q_{1111}, \dots, q_{0000})$ has the color that q_b has in $\mathcal{C}_b$. Thus, $\mathcal{G}'$ equipped with the parity condition of $\mathcal{G}^b$ is equivalent to $\mathcal{G}^b$.

To decide whether a strongly adaptive strategy exists, we proceed as follows:

1. We first give a characterization of the vertices v of $\mathcal{G}'$ with $\mathrm{smry}(v) = s$ that only uses summaries that are larger than s. This allows us to compute $\mathrm{smry}(v)$ for every vertex v by induction over the summaries.
2. Using the decomposition of $\mathcal{G}'$ into regions with the same summary and the characterization we construct a series of obliging games [14]. In an obliging game, Player 0 has a strong winning condition that has to be satisfied on every play and a weak winning condition that must be satisfiable if Player 1 cooperates. In our case, the strong winning condition requires Player 0 to always enforce the best value that is currently possible and the weak condition requires Player 1 to have a chance to make a bad move (if the summary encodes that this is still possible), i.e., whenever possible, Player 1 is given the chance to make a bad move.
3. Finally, if Player 1 has in all obliging games a strategy satisfying both the strong and the weak condition, then these can be turned effectively into a strongly adaptive strategy, otherwise there is no such strategy.

We first provide a useful lemma showing that a strategy in $\mathcal{G}'$ is strongly adaptive if and only if its summary is history independent, i.e., only depends on the last vertex.

Lemma 1. *A strategy σ for Player 0 in $\mathcal{G}'$ is strongly adaptive if and only if for every play prefix* $\mathtt{p}$ *ending in vertex v, it holds that* $\mathrm{smry}(\sigma, \mathtt{p})) = \mathrm{smry}(v)$.

As computed in Section 3.2, let $\mathrm{E}_{=b}$ be the set of vertices in $\mathcal{G}'$ from which the maximum value Player 0 can enforce is b. Furthermore, for a summary $s \in \mathcal{S}$, let $\mathrm{V}_{\geq s}$ denote the set of vertices v in $\mathcal{G}'$ for which $\mathrm{smry}(v) \geq_{\mathrm{lex}} s$. Let $\mathrm{V}_{=s}$, $\mathrm{V}_{>s}$, and $\mathrm{V}_{<s}$ be defined similarly.

Remark 9. Let $s = (b_0, \dots, b_k, \bot, \dots, \bot)$. Then, $\mathrm{V}_{=s} \subseteq \mathrm{E}_{=b_0}$.

For a vertex set F, let $\mathrm{pre}(F)$ denote the set of vertices from which there is an edge to F. Maybe surprisingly, we do not distinguish between vertices of Player 0 and Player 1, but we will only apply $\mathrm{pre}(F)$ when it is Player 1's turn.

Next, we characterize the sets $\mathrm{V}_{=s}$ in terms of the existence of strategies that witness summaries. The key aspects of this characterization is that it only refers to summaries $s' >_{\mathrm{lex}} s$, which will later allow us to compute these sets inductively.

Given a strategy σ for Player 0 in $\mathcal{G}'$ and a play prefix $\mathtt{p}$, let $\Pi(\sigma, \mathtt{p})$ denote the set of $(\sigma, \mathtt{p})$-plays that do not contain a bad move by Player 1 after $\mathtt{p}$.

Definition 3. *Let σ be a strategy for Player 0,* $\mathtt{p}$ *be a play prefix, and $s = (b_0, \dots, b_k, \bot, \dots, \bot)$ a summary. We say that σ is an s-witness from* $\mathtt{p}$ *if and only if it satisfies the following three properties:*

Enforcing *Every play in $\Pi(\sigma, \mathtt{p})$ satisfies the parity condition of the game $\mathcal{G}^{b_0}$. Thus, a witness has to enforce b_0 unless Player 1 makes a bad move.*

Enabling *If $k \geq 1$, there exists a play in $\Pi(\sigma, \mathtt{p})$ that visits $\mathrm{pre}\big(\mathrm{V}_{=\mathrm{lft}(s)}\big)$. Thus, if there is the chance to reach a vertex where Player 1 can make a bad move, then a witness has to visit such a vertex. Note that we require that the bad move leads to a vertex with summary $\mathrm{lft}(s)$, which is the largest summary that can be guaranteed to be reached from* $\mathtt{p}$ *after a bad move.*

Evading *If $k \geq 1$, then let us define $\mathrm{Ev}(s)$ to be the set of summaries $s' = (b'_0, \dots, b'_{k'}, \bot, \dots, \bot)$ with $b'_0 > b_0$, $s' <_{\mathrm{lex}} \mathrm{lft}(s)$, and such that s' is not a strict prefix of $\mathrm{lft}(s)$. Then, no play in $\Pi(\sigma, \mathtt{p})$ visits $\mathrm{pre}\big(\mathrm{V}_{=s'}\big)$ for any $s' \in \mathrm{Ev}(s)$. Thus, a witness can never reach a vertex where Player 1 can make a bad move to reach a summary that is worse than $\mathrm{lft}(s)$.*

Recall that $\mathrm{lft}(s) >_{\mathrm{lex}} s$ and that $s' \in \mathrm{Ev}(s)$ implies $s' >_{\mathrm{lex}} s$.

Our next lemma shows that witnesses do indeed witness the summaries of vertices.

Lemma 2. *In the game $\mathcal{G}'$, for some summary s and for some vertex v, we have $v \in \mathrm{V}_{=s}$ if and only if $v \notin \mathrm{V}_{>s}$ and there is an s-witness from v.*

We now give a method to compute $\mathrm{V}_{=s}$ for each summary $s \in \mathcal{S}$ by induction from the largest to the smallest summary. Since truth values in a summary are strictly increasing, $(1111, \bot, \dots, \bot)$ is the maximal summary. We have $\mathrm{V}_{=(1111, \bot, \dots, \bot)} = \mathrm{E}_{=1111}$, which we can compute using Tabuada and Neider's result for classical rLTL games (see Section 2). For the inductive step, assume that for a summary $s = (b_0, \dots, b_k, \bot, \dots, \bot)$ the sets $\mathrm{V}_{=s'}$ are already computed for every $s' >_{\mathrm{lex}} s$. The set $\mathrm{V}_{=s}$ can then be computed using the following algorithm:

1. If $k = 0$, then return $\mathrm{E}_{\geq b_0} \setminus \mathrm{V}_{>s}$. Here, $\mathrm{E}_{\geq b_0}$ can again be computed using Tabuada and Neider's result for classical rLTL games.
2. Now assume $k > 0$. Let $\mathcal{A}_s$ be the subgraph of $\mathcal{A}'$ restricted to the vertex set
$$\mathrm{E}_{\geq b_0} \setminus \Big(\mathrm{V}_{>s} \cup \bigcup_{s' \in \mathrm{Ev}(s)} \mathrm{Reach}_1(\mathrm{V}_{=s'})\Big),$$
where $\mathrm{Reach}_1(F)$ denotes the set of vertices of $\mathcal{A}'$ from which Player 1 can force the token to reach F. This set can be computed in linear time (in the number of edges of $\mathcal{A}'$) using standard methods to solve reachability games (see [18] for more details). In the proof of correctness of the algorithm (see Lemma 3) we show that $\mathcal{A}_s$ does not have any terminal vertices.
Also, the sets $\mathrm{V}_{=s'}$ for $s' \in \mathrm{Ev}(s)$ are already computed, because the summaries $s' \in \mathrm{Ev}(s)$ are all greater than s.
3. Let $\mathrm{Win}(s)$ be the winning region for Player 0 in the parity game with arena $\mathcal{A}_s$ and coloring as in the game $\mathcal{G}^{b_0}$. Return the set of vertices in Player 0's winning region $\mathrm{Win}(s)$ from which $\mathrm{pre}\big(\mathrm{V}_{=\mathrm{lft}(s)}\big)$ is reachable in the subgraph of $\mathcal{A}'$ restricted to $\mathrm{Win}(s)$.

Lemma 3. *The algorithm described above computes the sets* $\mathrm{V}_{=s}$ *for* $s \in \mathcal{S}$*.*

Now, we give a characterization of strongly adaptive strategies in terms of summary witnesses.

Lemma 4. *In the game* $\mathcal{G}'$*, a strategy* σ *is strongly adaptive if and only if it is a* $\mathrm{smry}(\mathtt{p})$*-witness from every play prefix* $\mathtt{p}$*.*

Now, we show how to decide whether a strategy satisfying the condition given in Lemma 4 exists, i.e., a strategy that is a $\mathrm{smry}(\mathtt{p})$-witness from every play prefix $\mathtt{p}$. Furthermore, if such a strategy exists, we compute one. To do so, we present a reduction to another type of game, called *obliging games*. So, before describing the details of the reduction, let us recapitulate the definitions and useful results on obliging games.

Obliging games are two-player games introduced by Chatterjee et al. [14]. They have two winning conditions, S and W, called strong and weak conditions. The objective of Player 0 is to ensure the strong winning condition while allowing Player 1 to cooperate with him to additionally fulfil the weak winning condition. Formally, a strategy σ for Player 0 is *uniformly gracious* if it satisfies the following:

– for every vertex v, every (σ, v)-play is S-winning, and
– for every play prefix $\mathtt{p}$ consistent with σ, there is a W-winning $(\sigma, \mathtt{p})$-play.

We are only interested in parity/Büchi obliging games (i.e., the strong condition is a parity condition, and the weak one is a Büchi condition). The next theorem follows directly from the results by Chatterjee et al. [14].

Theorem 3. *A parity/Büchi obliging game with n vertices and a parity condition with k colors can be reduced to a parity game with $O(n)$ vertices and $O(k)$ colors. Moreover, if Player 0 has a uniformly gracious strategy in such an obliging game, he has a uniformly gracious strategy with a memory of size at most $O(k)$.*

Now, coming back to our problem, we define obliging games $\mathcal{G}_s$ (for each $s \in \mathcal{S}$), which are subgames of $\mathcal{G}'$, such that a uniformly gracious strategy in $\mathcal{G}_s$ satisfies the properties of an s-witness locally. In particular, the games are defined in way such that the strong condition resembles the Enforcing property, the weak condition resembles the Enabling property, and the restricted vertex set ensures that the Evading property is satisfied.

Definition 4. *Given a summary $s = (b_0, \dots, b_k, \perp, \dots, \perp) \in \mathcal{S}$, let $\mathcal{G}_s$ be the obliging game obtained from $\mathcal{G}'$ as follows:*

- *The set of vertices $V(\mathcal{G}_s)$ is the set $\mathrm{V}_{=s} \cup \{v_{\mathrm{new}}\}$, where v_{new} is a new vertex that does not belong to V'.*
- *The set of edges $E(\mathcal{G}_s)$ contains the following edges:*
  - *The edges of the game $\mathcal{G}'$ restricted to the vertex set $\mathrm{V}_{=s}$.*
  - *All edges of the form (v, v_{new}) where v is a terminal vertex in the game $\mathcal{G}'$ restricted to $\mathrm{V}_{=s}$.*
  - *A self loop on v_{new}.*
- *The strong condition S_s is a min-parity condition such that the color of v_{new} is 0 and the color of any other vertex is the same as in $\mathcal{G}^{b_0}$.*
- *If $k = 0$, then there is no weak condition, i.e., W_s is a Büchi condition with $F = V(\mathcal{G}_s)$. If $k > 0$, then the weak condition W_s is a Büchi condition with $F = \mathrm{pre}(\mathrm{lft}(s)) \cup \{v_{\mathrm{new}}\}$.*

The following lemma formalizes the connection between uniformly gracious strategies in the obliging games $\mathcal{G}_s$ and strongly adaptive strategies in $\mathcal{G}'$.

Lemma 5. *There exists a strongly adaptive strategy in $\mathcal{G}'$ if and only if there exists a uniformly gracious strategy in every obliging game $\mathcal{G}_s$. Given a uniformly gracious strategy with finite memory in each obliging game $\mathcal{G}_s$, one can effectively combine these into a strongly adaptive strategy with finite memory in $\mathcal{G}'$.*

Since the game $\mathcal{G}'$ has doubly-exponential size, using Theorem 3, the parity/Büchi obliging games $\mathcal{G}_s$ can be reduced to doubly-exponential-sized parity games. Once we computed a strongly adaptive strategy for $\mathcal{G}'$, it can then be reduced to a strongly adaptive strategy for the original game $\mathcal{G}$.

Moreover, note that strongly adaptive strategies also have doubly-exponential memory since the obliging games we constructed have doubly-exponential size. By Theorem 3, uniformly gracious strategies in such obliging games require memory of linear size, leading to the following result.

Theorem 4. *Given an* rLTL *game, one can decide in doubly-exponential time whether Player 0 has a strongly adaptive strategy. If yes, one can compute one with doubly-exponential memory in doubly-exponential time.*

Note that by dualizing the definitions and the constructions, an analogous result for Player 1 can also be obtained.

5 Conclusion

We argued that in a reactive system, in addition to correctness, we also need to ensure robustness. To this end, we introduced adaptive strategies for rLTL games that satisfy the specification to a higher degree when the environment is not antagonistic. We also presented a stronger version of adaptive strategies that additionally maximizes the opportunities for the opponent to make bad choices. Finally, we showed that both adaptive and strongly adaptive strategies can be computed in doubly-exponential time. As we know that the classical LTL and rLTL synthesis algorithms also take doubly-exponential time, we conclude that adaptive and strongly adaptive strategies are not harder to compute.

In future work, we aim to investigate even more general notions of adaption to the behavior of a not necessarily antagonistic environment. Possible approaches include observing the environment's behavior and trying to compare that to optimal strategies for the environment.

References

1. Almagor, S., Kupferman, O.: Good-enough synthesis. In: Computer Aided Verification, CAV 2020, Part II. LLNCS, vol. 12225, pp. 541–563. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_28
2. Anevlavis, T., Neider, D., Phillipe, M., Tabuada, P.: Evrostos: the rLTL verifier. In: ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019. pp. 218–223. ACM (2019). https://doi.org/10.1145/3302504.3311812
3. Anevlavis, T., Philippe, M., Neider, D., Tabuada, P.: Verifying rLTL formulas: now faster than ever before! In: IEEE Conference on Decision and Control, CDC 2018. pp. 1556–1561. IEEE (2018). https://doi.org/10.1109/CDC.2018.8619014
4. Anevlavis, T., Philippe, M., Neider, D., Tabuada, P.: Being correct is not enough: Efficient verification using Robust Linear Temporal Logic. ACM Trans. Comput. Log. **23**(2), 8:1–8:39 (2022). https://doi.org/10.1145/3491216
5. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
6. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**(4) (sep 2011). https://doi.org/10.1145/2000799.2000800
7. Bloem, R., Chatterjee, K., Greimel, K., Henzinger, T.A., Hofferek, G., Jobstmann, B., Könighofer, B., Könighofer, R.: Synthesizing robust systems. Acta Informatica **51**(3-4), 193–220 (2014). https://doi.org/10.1007/s00236-013-0191-5
8. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Computer Aided Verification, CAV 2009. LLNCS, vol. 5643, pp. 140–156. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_14
9. Bloem, R., Ehlers, R., Jacobs, S., Könighofer, R.: How to handle assumptions in synthesis. In: Workshop on Synthesis, SYNT 2014. EPTCS, vol. 157, pp. 34–50 (2014). https://doi.org/10.4204/EPTCS.157.7
10. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: ACM SIGACT Symposium on Theory of Computing, STOC 2017. pp. 252–263. ACM (2017). https://doi.org/10.1145/3055399.3055409

11. Chatterjee, K., Doyen, L.: Energy parity games. Theor. Comput. Sci. **458**, 49–60 (2012). https://doi.org/10.1016/j.tcs.2012.07.038
12. Chatterjee, K., Henzinger, T.A.: Assume-guarantee synthesis. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2007, ETAPS 2007. LLNCS, vol. 4424, pp. 261–275. Springer (2007). https://doi.org/10.1007/978-3-540-71209-1_21
13. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Mean-payoff parity games. In: IEEE Symposium on Logic in Computer Science (LICS 2005). pp. 178–187. IEEE Computer Society (2005). https://doi.org/10.1109/LICS.2005.26
14. Chatterjee, K., Horn, F., Löding, C.: Obliging games. In: Gastin, P., Laroussinie, F. (eds.) Concurrency Theory, CONCUR 2010. LLNCS, vol. 6269, pp. 284–296. Springer (2010). https://doi.org/10.1007/978-3-642-15375-4_20
15. Dallal, E., Neider, D., Tabuada, P.: Synthesis of safety controllers robust to unmodeled intermittent disturbances. In: IEEE Conference on Decision and Control, CDC 2016. pp. 7425–7430. IEEE (2016). https://doi.org/10.1109/CDC.2016.7799416
16. Ehlers, R., Topcu, U.: Resilience to intermittent assumption violations in reactive synthesis. In: International Conference on Hybrid Systems: Computation and Control, HSCC'14. pp. 203–212. ACM (2014). https://doi.org/10.1145/2562059.2562128
17. Fearnley, J., Zimmermann, M.: Playing Muller games in a hurry. Int. J. Found. Comput. Sci. **23**(3), 649–668 (2012). https://doi.org/10.1142/S0129054112400321
18. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], LLNCS, vol. 2500. Springer (2002). https://doi.org/10.1007/3-540-36387-4
19. Kuhn, H.W.: Extensive games and the problem of information. Princeton University Press, Princeton, NJ (1953)
20. Majumdar, R., Render, E., Tabuada, P.: A theory of robust omega-regular software synthesis. ACM Trans. Embed. Comput. Syst. **13**(3), 48:1–48:27 (2013). https://doi.org/10.1145/2539036.2539044
21. Mascle, C., Neider, D., Schwenger, M., Tabuada, P., Weinert, A., Zimmermann, M.: From LTL to rLTL monitoring: improved monitorability through robust semantics. In: HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control. pp. 7:1–7:12. ACM (2020). https://doi.org/10.1145/3365365.3382197
22. Nayak, S.P., Neider, D., Roy, R., Zimmermann, M.: Robust computation tree logic. In: Deshmukh, J.V., Havelund, K., Perez, I. (eds.) NASA Formal Methods - 14th International Symposium, NFM 2022, Pasadena, CA, USA, May 24-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13260, pp. 538–556. Springer (2022). https://doi.org/10.1007/978-3-031-06773-0_29
23. Nayak, S.P., Neider, D., Zimmermann, M.: Adaptive strategies for rLTL games. In: HSCC '21: ACM International Conference on Hybrid Systems: Computation and Control. pp. 32:1–32:2. ACM (2021). https://doi.org/10.1145/3447928.3457210
24. Nayak, S.P., Neider, D., Zimmermann, M.: Robustness-by-construction synthesis: Adapting to the environment at runtime. CoRR **abs/2204.10912** (2022). https://doi.org/10.48550/arXiv.2204.10912
25. Neider, D., Totzke, P., Zimmermann, M.: Optimally resilient strategies in pushdown safety games. In: International Symposium on Mathematical Foundations of Computer Science, MFCS 2020. LIPIcs, vol. 170, pp. 74:1–74:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.MFCS.2020.74
26. Neider, D., Weinert, A., Zimmermann, M.: Synthesizing optimally resilient controllers. In: EACSL Annual Conference on Computer Science Logic, CSL 2018.

LIPIcs, vol. 119, pp. 34:1–34:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.CSL.2018.34

27. Neider, D., Weinert, A., Zimmermann, M.: Robust, expressive, and quantitative linear temporal logics: Pick any two for free. Information and Computation p. 104810 (2021). https://doi.org/https://doi.org/10.1016/j.ic.2021.104810
28. Pnueli, A.: The temporal logic of programs. In: Symposium on Foundations of Computer Science, 1977. pp. 46–57. IEEE Computer Society (1977). https://doi.org/10.1109/SFCS.1977.32
29. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: ACM Symposium on Principles of Programming Languages, 1989. pp. 179–190. ACM Press (1989). https://doi.org/10.1145/75277.75293
30. Pnueli, A., Rosner, R.: On the synthesis of an asynchronous reactive module. In: Automata, Languages and Programming, ICALP89. LLNCS, vol. 372, pp. 652–671. Springer (1989). https://doi.org/10.1007/BFb0035790
31. Priest, G.: Dualising intuitionictic negation. Principia: an international journal of epistemology **13**(2), 165–184 (2009). https://doi.org/10.5007/1808-1711.2009v13n2p165
32. Samuel, S., Mallik, K., Schmuck, A., Neider, D.: Resilient abstraction-based controller design. In: HSCC '20: ACM International Conference on Hybrid Systems: Computation and Control. pp. 33:1–33:2. ACM (2020). https://doi.org/10.1145/3365365.3383467
33. Samuel, S., Mallik, K., Schmuck, A., Neider, D.: Resilient abstraction-based controller design. In: IEEE Conference on Decision and Control, CDC 2020. pp. 2123–2129. IEEE (2020). https://doi.org/10.1109/CDC42340.2020.9303932
34. Schewe, S., Varghese, T.: Tight bounds for the determinisation and complementation of generalised Büchi automata. In: Automated Technology for Verification and Analysis, ATVA 2012. LLNCS, vol. 7561, pp. 42–56. Springer (2012). https://doi.org/10.1007/978-3-642-33386-6_5
35. Tabuada, P., Caliskan, S.Y., Rungger, M., Majumdar, R.: Towards robustness for cyber-physical systems. IEEE Trans. Autom. Control. **59**(12), 3151–3163 (2014). https://doi.org/10.1109/TAC.2014.2351632
36. Tabuada, P., Neider, D.: Robust linear temporal logic. In: Conference on Computer Science Logic, CSL 2016. LIPIcs, vol. 62, pp. 10:1–10:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016). https://doi.org/10.4230/LIPIcs.CSL.2016.10
37. Topcu, U., Ozay, N., Liu, J., Murray, R.M.: On synthesizing robust discrete controllers under modeling uncertainty. In: Hybrid Systems: Computation and Control, HSCC'12. pp. 85–94. ACM (2012). https://doi.org/10.1145/2185632.2185648
38. Ummels, M.: Rational behaviour and strategy construction in infinite multiplayer games. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4337, pp. 212–223. Springer (2006). https://doi.org/10.1007/11944836_21, https://doi.org/10.1007/11944836_21