MASTER THESIS

Adaptive Strategies for rLTL Games

Author:
Satya Prakash NAYAK

Supervisors:
Dr. Daniel NEIDER
Dr. Martin ZIMMERMANN

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Department of Computer Science
Chennai Mathematical Institute

May 29, 2021

Declaration of Authorship

I, Satya Prakash NAYAK, declare that this thesis titled, "Adaptive Strategies for rLTL Games" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

We consider the problem of synthesizing the controllers that are optimal with respect to a quality criterion based on Robust Linear Temporal Logic (rLTL). We address such problems by formulating it as a two-player game, called rLTL game. We show that the current algorithms for rLTL games do not compute optimal controllers. They assume the environment to always act antagonistically, which is often a non-realistic assumption. So is there a way of satisfying the specification ”better” if the environment is not antagonistic? In order to solve this inefficiency, we develop two new notions of strategies. The first notion is that of Adaptive strategies, which, in response to the opponent's bad choices (i.e., suboptimal choices), adapts to ensure optimality w.r.t. the current stage. The second notion is that of Strongly adaptive strategies, which is a stronger version of the first one that also maximizes the opportunities for the opponent to make bad choices. We show that the computability problem for both types of strategies is not harder than the classical one and can be solved in doubly-exponential time.

Acknowledgements

First and foremost, I wish to express my deepest gratitude to my supervisors Dr. Daniel Neider and Dr. Martin Zimmermann, for their invaluable guidance and constant support and encouragement. I am thankful to my supervisors for providing me the opportunity to work on this project during this unprecedented pandemic.

I am grateful to the Computer Science department at Chennai Mathematical Institute for the exciting courses and curriculum which introduced me to the field of Logic, Games and Automata Theory. I would particularly like to thank Dr. B Srivathsan, Dr. Aiswarya Cyriac, and Dr. Prajakta Nimbhorkar for the wonderful courses they offered in Automata and Algorithms.

Finally, I wish to thank my family and friends for their unfailing support and continuous encouragement throughout all these years.

Contents

List of Figures

List of Symbols

$\mathbb{N}$	set of Natural numbers $\{1, 2, 3, \ldots\}$	i, j, k, n
$\mathbb{B}$	set of LTL Truth values $\{1, 0\}$	
$\mathbb{B}_4$	set of rLTL Truth values $\{1111, 0111, 0011, 0001, 0000\}$	a, b
$\emptyset$	empty set $\{\}$	
$\mathcal{P}$	set of atomic propositions	p, q
φ, ψ	logical formulas	
w	infinite word in $(2^{\mathcal{P}})^{\omega}$	w_i
$w(i)$	i-th symbol of w	
$w_{i\ldots}$	suffix of w starting at position i	
V	Valuation function	
cl	closure (set of subfomulas)	
$\mathcal{B}$	Büchi automaton	
$\mathcal{C}$	Parity automaton	
Q	set of states	q, q_i
q_0	initial state in Q	
Σ	set of alphabets	
ρ	a run/play	
δ/Δ	deterministic/non-deterministic transition relation	
F	set of final/accepting states	
$\mathcal{F}$	set of accepting sets of states	F
Ω	priority function	
$\mathfrak{G}$	A game	
$\text{win}(\mathfrak{G})$	winning region for Player 0 in game $\mathfrak{G}$	
$\mathcal{G}$	Game graph	
V	set of vertices	
V_0	set of Player 0's vertices	
V_1	set of Player 1's vertices	
E	edge relation	
λ	label function $V \to 2^{\mathcal{P}}$	
$P^n/P^*/P^{\omega}$	set of n-length/finite/infinite paths in a graph	ρ, p
ρ	a play/ an infinite path	
p	a play prefix/ a finite path	
σ, τ	strategies for a game	

Chapter 1

Introduction

Nowadays, formal methods are being used extensively in the verification of safety-critical software and hardware. It has been used to verify air traffic control, autopilots, medical equipment, CPU designs, and many other systems. Moreover, temporal logics are being applied to formal methods in computer science since 1970. Now temporal logics are widely used for specifying complex reactive (i.e., dynamic, concurrent, distributed) systems. In this thesis, we address the problem of synthesizing optimal controllers for the systems with specifications expressed in Robust Linear Temporal Logic (rLTL).

Robust LTL was introduced by Tabuada and Neider, 2016 to capture the concept of robustness in temporal logics. The difference between "minor" and "major" violations of a formula cannot be distinguished in a two-valued semantics. For example, consider the formula $\varphi = \Box p$ for some atomic proposition 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 in the following order:

$$1111 \succ 0111 \succ 0011 \succ 0001 \succ 0000.$$

Intuitively, 1111 corresponds to true, and the rest to different shades of false. For the above example, consider the robust version of the formula, i.e., 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. For example, consider a game played between two players: Player 0 (controller) and Player 1 (environment), as shown in Figure 1.1. Player 1's vertices are shown as squares, and Player 0's vertices are shown as circles. Each vertex is labeled by a set of propositions, e.g., Vertex 4 is labeled by propositions p and q.

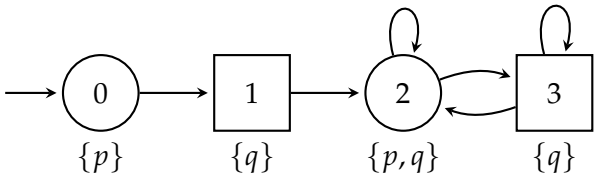

FIGURE 1.1: An rLTL game

Suppose a token is initially placed at Vertex 0. At any stage, if the token is in a vertex of Player i, $i \in \{0, 1\}$, 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 $\{p\}\{q\}\{p,q\}\{q\}\{p,q\}\{q\}\ldots$ Suppose the rLTL specification is $\boxdot p$ for Player 0, 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., 0122...) 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 0'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 1 is to minimize the value of the formula on any play.

However, the classical controllers computed by Tabuada and Neider are not optimally robust. It only performs the worst-case analysis and assumes that the environment always makes the best moves possible, which is not very realistic. In order to solve this inefficiency, we introduce two new notions of strategies: Adaptive strategies and Strongly adaptive strategies. The first one is a strategy that adapts its moves to ensure optimality once the environment has made a bad move (by "bad", we mean the moves which are not optimal). We give an algorithm that computes an adaptive strategy for a player in an rLTL game in doubly-exponential time by reducing the problem to solving Parity games (Calude et al., 2017). As we know that the currently known algorithms for the classical synthesis problems for the controllers with an LTL specification also take doubly-exponential time, we conclude that computing adaptive strategies are not harder than computing the standard ones.

The second notion is a stronger version of the first one that also maximizes the chances of the environment making bad moves. However, such a strategy may not even exist in some cases. We show that the existence of a strongly adaptive strategy can be decided in doubly-exponential time. Moreover, we give a doubly-exponential time algorithm that computes a strongly adaptive strategy (if one exists at all) by reducing it to a series of Parity games and Obliging games (Chatterjee, Horn, and Löding, 2010). Hence, computing a strongly adaptive strategy is also not harder than the classical synthesis problems.

Chapter 2

Robust Linear Temporal Logic

In this chapter, we describe the syntax and semantics of Robust LTL and how it is different from classical LTL. We then show that LTL and rLTL are equally expressive. Moreover, we show an effective translation from LTL to rLTL formulas and vice versa. Finally, for a given rLTL formula φ and a set of values B, we show that a generalized Büchi automaton recognizing all the infinite words on which the value of φ belongs to B can be constructed.

In reactive systems, the specifications are usually of the form $\varphi \Rightarrow \psi$. The specification φ represents the environment assumption under which the system guarantees ψ needs to be ensured. However, the specification is equivalent to $\neg\varphi \vee \psi$ in LTL, which means the system can behave arbitrarily even when the environment assumption is minorly violated. For example, consider the specifications $\varphi = \Box p$ and $\psi = \Box q$ for some atomic propositions p, q. Then the specification $\Box p \Rightarrow \Box q$ is satisfied even if p does not hold only at a single position and q does not hold at any position. Hence, Tabuada and Neider introduced a robust version of LTL, called Robust LTL, in which a small violation of φ leads to (at most) a small violation of ψ. We will see how rLTL preserves robustness once we have described the syntax and semantics of it.

2.1 Syntax

We fix some finite non-empty set $\mathcal{P}$ of atomic propositions. The syntax of rLTL is similar to the syntax of LTL with the only difference being the use of dotted temporal operators. More precisely, the 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 $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$, $\odot\varphi$, $\mathbf{R}\,\varphi$ and $\mathbf{U}\,\varphi$; where the temporal operators $\odot$, $\mathbf{R}$, and $\mathbf{U}$ correspond to "next", "release" and "until", respectively.

Additional temporal operators are $\boxdot\varphi$ and $\diamondsuit\varphi$, which correspond to "always" and "eventually", respectively, and these can be recovered from the operators $\mathbf{U}$ and

R as follows:

$$\boxdot \varphi = \mathit{false}\ \mathbf{R}\ \varphi$$
$$\diamondsuit \varphi = \mathit{true}\ \mathbf{U}\ \varphi$$

The conjunction operator $\vee$ and implication operator $\Rightarrow$ can be derived from negation $\neg$ and disjunction $\wedge$ in LTL. Similarly, the temporal operator **R** can also be derived from the until operator **U** and negation $\neg$. However, this is not possible in rLTL since it is a 5-valued semantics. Therefore, these operators are directly included in the syntax definition.

2.2 Semantics

For an infinite word $w = w_0 w_1 \ldots \in (2^{\mathcal{P}})^\omega$ and $i \in \mathbb{N}$, let $w(i) = w_i$ denotes the i-th symbol of w; and $w_{i..} = w_i w_{i+1} \ldots$ denotes the (infinite) suffix of w starting at position i.

The rLTL semantics is a mapping V, called *valuation*, that maps an infinite word $w \in 2^{\mathcal{P}}$ and an rLTL formula φ to an element of $\mathbb{B}_4$. For $1 \leq k \leq 4$, let $V_k(w, \varphi)$ be the kth entry of $V(w, \varphi)$, i.e.,

$$V(w, \varphi) = \Big(V_1(w, \varphi), V_2(w, \varphi), V_3(w, \varphi), V_4(w, \varphi)\Big).$$

Then V is inductively defined as follows.

- For an atomic proposition $p \in \mathcal{P}$, it is defined classically as in LTL:

  (1) $V(w, p) = \begin{cases} 0000 & \text{if } p \notin w(0) \\ 1111 & \text{if } p \in w(0) \end{cases}$

- For logical connectives, it is defined using *da costa algebra* (Priest, 2009):

  (2) $V(w, \neg\varphi) = \overline{V(w, \varphi)}$; where $\overline{a} = \begin{cases} 0000 & \text{if } a = 1111 \\ 1111 & \text{otherwise} \end{cases}$

  (3) $V(w, \varphi \vee \psi) = \max\Big\{V(w, \varphi), V(w, \psi)\Big\}$

  (4) $V(w, \varphi \wedge \psi) = \min\Big\{V(w, \varphi), V(w, \psi)\Big\}$

  (5) $V(w, \varphi \Rightarrow \psi) = V(w, \varphi) \to V(w, \psi)$; where $a \to b = \begin{cases} 1111 & \text{if } a \preceq b \\ b & \text{otherwise} \end{cases}$

  Note that the negation operator treats 1111 as *true* and other values as (different shades of) *false* as we have discussed in Chapter 1, i.e., it maps 1111 to *false* and other values to *true* in a sense.

- For temporal operators, it is a generalization of LTL semantics:

  (6) $V(w, \odot\varphi) = V(w_{1..}, \varphi)$

(7) $V(w, \varphi \mathbf{R} \psi) = (\inf_{j\geq 0} W_1, \sup_{k\geq 0} \inf_{j\geq k} W_2, \inf_{k\geq 0} \sup_{j\geq k} W_3, \sup_{j\geq 0} W_4)$; where

$$W_k = \max\left(V_k(w_{j..}, \psi), \sup_{0\leq i<j} V_k(w_{i..}, \varphi)\right)$$

(8) $V(w, \varphi \mathbf{U} \psi)$ is defined such that

$$V_k(w, \varphi \mathbf{U} \psi) = \sup_{j\geq 0} \min\left(V_k(w_{j..}, \psi), \inf_{0\leq i<j} V_k(w_{i..}, \varphi)\right)$$

- Semantics for additional temporal operators can also be generalized as follows:

$$(9)\ V(w, \boxdot \varphi) = \left(\inf_{i\geq 0} V_1(w_{i..}, \varphi), \sup_{j\geq 0} \inf_{i\geq j} V_2(w_{i..}, \varphi), \inf_{j\geq 0} \sup_{i\geq j} V_3(w_{i..}, \varphi), \sup_{i\geq 0} V_4(w_{i..}, \varphi)\right)$$

$$(10)\ V(w, \diamondsuit \varphi) = \left(\sup_{i\geq 0} V_1(w_{i..}, \varphi), \sup_{i\geq 0} V_2(w_{i..}, \varphi), \sup_{i\geq 0} V_3(w_{i..}, \varphi), \sup_{i\geq 0} V_4(w_{i..}, \varphi)\right)$$

Then we can see that for the formula $\boxdot p$, the valuation $V(w, \boxdot p)$ can be expressed in terms of LTL valuation W by

$$V(w, \boxdot p) = \Big(W(w, \Box p), W(w, \diamondsuit \Box p), W(w, \Box \diamondsuit p), W(w, \diamondsuit p)\Big).$$

This evaluates to different values in $\mathbb{B}_4$ distinguishing various degree of violations as we have seen in Chapter 1. Now consider the specification $\boxdot p \Rightarrow \boxdot q$ as we have seen at the beginning of this chapter and assume it evaluates to 1111 for some infinite word. Let us see how the system behaves in response to various degrees of violation of environment assumption.

- If p holds at all positions, then $\boxdot p$ evaluates to 1111, hence by Equation (3), $\boxdot q$ 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 $\boxdot p$ is minorly violated and p holds eventually always, then $\boxdot p$ evaluates to 0111 and then by Equation (3), $\boxdot q$ evaluates to 0111 or higher. Hence, 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. 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 guaranty is not satisfied. However, the value of $\boxdot p \Rightarrow \boxdot q$ still describes which weakened guarantee follows from the environment assumption.

2.3 Expressivenes

As we know that, rLTL is an extension of LTL. Moreover, in this section, we show that the LTL semantics of a formula containing no implication can be recovered from the first bit of the rLTL semantics. Conversely, each bit of the valuation function for an rLTL formula φ can be expressed by evaluating an LTL formula φ_i, obtained by a straightforward rewriting of the formula φ. Hence, we conclude the following result.

Theorem 2.1. LTL *and* rLTL *are equally expressive. Moreover, the translation from* LTL *to* rLTL *and vice versa are effective.*

A direct corollary is that for any decidable problem for LTL, the corresponding problem for rLTL is also decidable.

2.3.1 From LTL to rLTL

Recall that $V_k(w, \varphi)$ represents the kth bit of the rLTL valuation $V(w, \varphi)$. Let $\pi_k : \mathbb{B}_4 \to \mathbb{B}$ be a function defined by $\pi_k(a_1, a_2, a_3, a_4) = a_k$. Then we can see that V_1 is actually the composite function $\pi_1 \circ V$. Then we show that V_1 is indeed an LTL valuation for a formula containing no implication.

- For an atomic proposition $p \in \mathcal{P}$:

  (1) $V_1(w, p) = \begin{cases} \pi_1(0000) = 0 & \text{if } p \notin w(0) \\ \pi_1(1111) = 1 & \text{if } p \in w(0) \end{cases}$

- For logical connectives, we can evaluate in the following way:

  (2) $V_1(w, \neg\varphi) = \pi_1\Big(\overline{V(w, \varphi)}\Big) = 1 - V_1(w, \varphi)$

  (3) $V_1(w, \varphi \vee \psi) = \pi_1\Big(\max\big(V(w, \varphi), V(w, \psi)\big)\Big) = \max\Big(V_1(w, \varphi), V_1(w, \psi)\Big)$

  (4) $V_1(w, \varphi \wedge \psi) = \pi_1\Big(\min\big(V(w, \varphi), V(w, \psi)\big)\Big) = \min\Big(V_1(w, \varphi), V_1(w, \psi)\Big)$

- For temporal operators, it directly follows from the semantics:

  (5) $V_1(w, \odot\varphi) = V_1(w_{1..}, \varphi)$

  (6) $V_1(w, \varphi\, \mathbf{R}\, \psi) = \inf_{j\geq 0} \max\Big(V_1(w_{j..}, \psi), \sup_{0\leq i<j} V_1(w_{i..}, \varphi)\Big)$

  (7) $V_1(w, \varphi\, \mathbf{U}\, \psi) = \sup_{j\geq 0} \min\Big(V_1(w_{j..}, \psi), \inf_{0\leq i<j} V_1(w_{i..}, \varphi)\Big)$

  (8) $V_1(w, \boxdot\varphi) = \inf_{i\geq 0} V_1(w_{i..}, \varphi)$

  (9) $V_1(w, \diamondsuit\varphi) = \sup_{i\geq 0} V_1(w_{i..}, \varphi)$

Note that $V_1(w, \varphi) = 1$ if and only if $V(w, \varphi) = 1111$.

As we know that $\varphi \Rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$ in LTL, we can rewrite any LTL formula into a formula containing no implication. Suppose W be the LTL valuation.

Then, given an LTL formula φ, we can construct an rLTL formula φ' such that for any infinite word w, we have

$$W(w, \varphi) = 1 \text{ if and only if } V(w, \varphi') = 1111.$$

Therefore, we have shown that rLTL is as expressive as LTL.

2.3.2 From rLTL to LTL

To show that LTL is also as expressive as rLTL, given an rLTL formula φ, we construct four maps L_k for all $1 \leq k \leq 4$ that maps each rLTL φ formula to an LTL formula such that for any infinite word w, we have

$$V_k(w, \varphi) = W(w, L_k(\varphi)) \quad \text{for all } 1 \leq k \leq 4.$$

The maps $L_k : 1 \leq k \leq 4$ can be constructed inductively as follows:

(1) $L_k(p) = p$ for every atomic proposition p and for all $1 \leq k \leq 4$.

(2) $L_k(\neg\varphi) = \neg L_1(\varphi)$ for all $1 \leq k \leq 4$.

(3) $L_k(\varphi \vee \psi) = L_k(\varphi) \vee L_k(\psi)$ for all $1 \leq k \leq 4$.

(4) $L_k(\varphi \wedge \psi) = L_k(\varphi) \wedge L_k(\psi)$ for all $1 \leq k \leq 4$.

(5) $L_4(\varphi \Rightarrow \psi) = L_4(\varphi) \Rightarrow L_4(\psi)$; and
$L_k(\varphi \Rightarrow \psi) = \big(L_k(\varphi) \Rightarrow L_k(\psi)\big) \wedge L_{k+1}(\varphi \Rightarrow \psi)$ for all $1 \leq k \leq 3$.

(6) $L_k(\odot\varphi) = \bigcirc L_k(\varphi)$ for all $1 \leq k \leq 4$.

(7) $L_1(\varphi \,\mathbf{R}\, \psi) = L_1(\varphi)\,\mathbf{R}\,L_1(\psi)$;
$L_2(\varphi \,\mathbf{R}\, \psi) = \Diamond\Box L_2(\varphi) \vee \Diamond L_2(\psi)$;
$L_3(\varphi \,\mathbf{R}\, \psi) = \Box\Diamond L_3(\varphi) \vee \Diamond L_3(\psi)$;
$L_4(\varphi \,\mathbf{R}\, \psi) = \Diamond L_4(\varphi) \vee \Diamond L_4(\psi)$.

(8) $L_k(\varphi \,\mathbf{U}\, \psi) = L_k(\varphi)\,\mathbf{U}\,L_k(\psi)$ for all $1 \leq k \leq 4$.

(9) $L_1(\boxdot\varphi) = \Box L_1(\varphi)$; $\quad L_2(\boxdot\varphi) = \Diamond\Box L_2(\varphi)$;
$L_3(\boxdot\varphi) = \Box\Diamond L_3(\varphi)$; $\quad L_4(\boxdot\varphi) = \Diamond L_4(\varphi)$.

(10) $L_k(\diamondsuit\varphi) = \Diamond L_k(\varphi)$ for all $1 \leq k \leq 4$.

It is easy to verify that the function has the desired property indeed.

2.3.3 From rLTL to Generalized Büchi Automata

We know that for any LTL formula, a (generalized) Büchi automaton can be constructed that recognizes the infinite words satisfying the formula. A similar Büchi automaton can also be constructed for rLTL formulas. Before going into details, let us first briefly summarize the definition of Büchi automata.

Definition 2.1 (Büchi Automata)**.** A non-deterministic *Büchi automaton* is a tuple $\mathcal{B} = (Q, \Sigma, q_0, \Delta, F)$ consisting of a finite, non-empty set Q of states, a finite alphabet Σ, an initial state $q_0 \in Q$, a transition function $\Delta \subseteq Q \times \Sigma \to Q$, and an accepting set $F \subseteq Q$.

The *run* of the non-deterministic Büchi automaton $\mathcal{B}$ on a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ is an infinite sequence of states $\rho = q_0 q_1 \ldots \in Q^\omega$ starting at q_0 and satisfying $(q_i, w_i, q_{i+1}) \in \Delta$ for all $i \in \mathbb{N}$. We denote the set of states occurring infinitely often during ρ by $\mathrm{Inf}(\rho) = \{q \in Q \mid \forall i \in \mathbb{N}, \exists j \geq i \text{ such that } q_j = q\}$. A run ρ is called *accepting* if the run visits a state of F infinitely often, i.e., $\mathrm{Inf}(\rho) \cap F \neq \emptyset$. The language of a Büchi automaton $\mathcal{B}$, denoted by $L(\mathcal{B})$, is the set of all infinite words for which an accepting run of $\mathcal{B}$ exists.

A *generalized Büchi automaton* $\mathcal{B} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ is a variant of Büchi automaton with only difference being its accepting condition, i.e., a set of accepting sets $\mathcal{F} \subseteq 2^Q$. A run ρ is accepting if it visits a state of every set $F \in \mathcal{F}$ infinitely often, i.e., $\mathrm{Inf}(\rho) \cap F \neq \emptyset$ for all $F \in \mathcal{F}$.

The translation of LTL formulas into generalized Büchi automata is based on the expansion rules, by which it tracks the value of a given formula and its subformulas at each position of an infinite word. The size of such automata depends on the closure of that formula, which is formally defined next.

Definition 2.2 (closure)**.** The closure of an rLTL formula φ, denoted by $\mathrm{cl}(\varphi)$, is inductively defined as follows:

(1) $\mathrm{cl}(p) = \{p\}$

(2) $\mathrm{cl}(\neg\varphi) = \{\neg\varphi\} \cup \mathrm{cl}(\varphi)$

(3) $\mathrm{cl}(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$

(4) $\mathrm{cl}(\varphi \wedge \psi) = \{\varphi \wedge \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$

(5) $\mathrm{cl}(\varphi \Rightarrow \psi) = \{\varphi \Rightarrow \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$

(6) $\mathrm{cl}(\odot\varphi) = \{\odot\varphi\} \cup \mathrm{cl}(\varphi)$

(7) $\mathrm{cl}(\varphi \,\mathbf{R}\, \psi) = \{\varphi \,\mathbf{R}\, \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$

(8) $\mathrm{cl}(\varphi \,\mathbf{U}\, \psi) = \{\varphi \,\mathbf{U}\, \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$

(9) $\mathrm{cl}(\boxdot\varphi) = \{\boxdot\varphi\} \cup \mathrm{cl}(\varphi)$

(10) $\mathrm{cl}(\diamondsuit\varphi) = \{\diamondsuit\varphi\} \cup \mathrm{cl}(\varphi)$

Given an LTL formula φ, the construction of translation of LTL into Büchi automata given by Baier and Katoen, 2008, results in an automaton of size $O(|\mathrm{cl}(\varphi)| \cdot 2^{|\mathrm{cl}(\varphi)|})$. Since rLTL has 5-valued semantics, following a similar method, Tabuada and Neider, 2016, have shown that, given an rLTL formula φ, a similar Büchi automaton of size $O(|\mathrm{cl}(\varphi)| \cdot 5^{|\mathrm{cl}(\varphi)|})$ can be constructed. Formally, the following result has been obtained.

Theorem 2.2. *Given an* rLTL *formula* φ *and a set of values* $B \subseteq \mathbb{B}_4$*, one can construct a generalized Büchi automaton* $\mathcal{B}_\varphi$ *with* $5^{|cl(\varphi)|} + 5$ *states and* $4 \cdot |cl(\varphi)|$ *accepting sets that recognizes the set of infinite words on which the value of* φ *belongs to* B*, i.e.,*$L(\mathcal{B}_\varphi) = \{w \in (2^{\mathcal{P}})^\omega : V(w, \varphi) \in B\}$.

Complexity results for model checking and reactive synthesis of rLTL follow from this construction. We also use this construction for the synthesis of adaptive strategies in the next chapter.

Chapter 3

rLTL Games

In this chapter, we present a slightly modified version of the rLTL game introduced by Tabuada and Neider, 2016. We motivate our work with a few examples of games. We analyze what a classical strategy does in an rLTL game and why that is not optimal. Then we present adaptive strategies and describe why these are better than the classical ones. Furthermore, we also present strongly adaptive strategies and describe their importance and existence. We also show that both types of strategies can be computed in doubly-exponential time and hence are not harder than the classical ones.

3.1 Definitions and Notations

We consider infinite-duration two-player games over finite graphs with rLTL specifications as introduced by Tabuada and Neider, 2016 with a minor modification, where Player 0 with given specification φ prefers the value 1111 over 0111 and prefers the value 0111 over 0011 and so on. Intuitively, if a player has a winning condition φ, he will try his best to get a play that satisfies the formula φ. If that is not possible, he will try his best to satisfy a violated version of the formula φ. For example, if $\varphi = \boxdot p$ for some proposition p, then the player will try to satisfy $\square p$, and if that is not possible for this game, he will try to satisfy $\diamondsuit\square p$. If that is also not possible, then he will try to satisfy $\square\diamondsuit p$ and so on.

Definition 3.1 (rLTL Games)**.** An *rLTL game* is pair $\mathfrak{G} = (\mathcal{G}, \varphi)$ consisting of

- a finite, labeled game graph $\mathcal{G} = (V, E, \lambda)$ where V is a finite set of vertices that is partitioned into two disjoint sets $V_0, V_1 \subseteq V$, $E \subseteq V \times V$ is a directed edge relation, and $\lambda : V \to 2^{\mathcal{P}}$ is a function labelling each vertex with atomic propositions; and
- an rLTL formula φ over $\mathcal{P}$.

For $n \geq 0$, let P^n denote the set of paths in $\mathcal{G}$ of length n (in particular $P^0 = V, P^1 = E$). P^* will denote the set of all finite paths in $\mathcal{G}$ and P^ω the set of all infinite paths.

An rLTL game is played by two players, Player 0 and Player 1, who construct a *play* $\rho = v_0 v_1 \ldots \in P^\omega$ by moving a token along the edges of the game graph. A play

$\rho = v_0 v_1 \ldots$ induces an infinite word $\lambda(\rho) = \lambda(v_0)\lambda(v_1)\ldots \in (2^{\mathcal{P}})^\omega$, and the *value* of the play, $V(\rho)$ is the value of the formula φ on $\lambda(\rho)$. Player 0's objective is to maximize the value, while Player 1's objective is to minimize it.

Strategies

A *strategy* for Player i, $i \in \{0,1\}$, is a function $\sigma : P^* \cap V^* V_i \to V$, which assigns to each possible finite path αu ending in $u \in V_i$ a neighbour $\sigma(\alpha u) \in N(u)$, where $N(u) = \{v : (u,v) \in E\}$ is the set of outgoing neighbours of u. It prescribes the next move of Player i depending on the finite play played thus far.

For a strategy σ of Player i, a finite or an infinite play $\rho = v_0 v_1 \ldots$ is said to be a *σ-play*, if whenever $v_j \in V_i$, $v_{j+1} = \sigma(v_0 v_1 \ldots v_j)$.

A *play prefix* or a finite play is a finite path $\mathtt{p} \in P^*$ in the graph. For a play prefix $\mathtt{p} = v_0 v_1 \ldots v_n$ and a strategy σ for Player i, any finite or infinite play $\rho = \mathtt{p} v_{n+1} v_{n+2} \ldots$ is said to be a *σ-continuation* of $\mathtt{p}$ if it satisfies the following: if $v_k \in V_i$ then $v_{k+1} = \sigma(v_0 v_1 \ldots v_k)$ for all $k \geq n$. Note that the prefix $\mathtt{p}$ does not have to be consistent with the strategy σ.

In the original paper on robust LTL by Tabuada and Neider, 2016, a doubly-exponential time algorithm is given that solves the classical rLTL synthesis problem, which is equivalent to solving the following problem.

Problem 3.1. Given an rLTL game $\mathfrak{G}$ with an initial vertex v_0 and a value b, synthesize a strategy σ (if one exists) for each player such that every σ-play has value b.

3.2 Adaptive Strategies

In this section, we start by presenting a motivating example and a classical strategy for Player 0 in that game. We describe why that strategy is not optimal and what would be a better one. We then formalize the definition of adaptive strategies and give a doubly-exponential algorithm to compute it.

3.2.1 Motivating Example

Consider the game arena $\mathcal{G}$ in the Figure 3.1 (Player 1's vertices are shown as squares and Player 0's vertices are shown as circles) with rLTL specification $\varphi = \boxdot p$ as shown in the figure.

Suppose the token is initially placed at Vertex 0. Considering Player 1 plays his best moves, the best possible scenario for Player 0 in this game 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 his vertices: $\{0 \to 1; 3 \to 2; 4 \to 1\}$. As we can see that, if 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 eventually

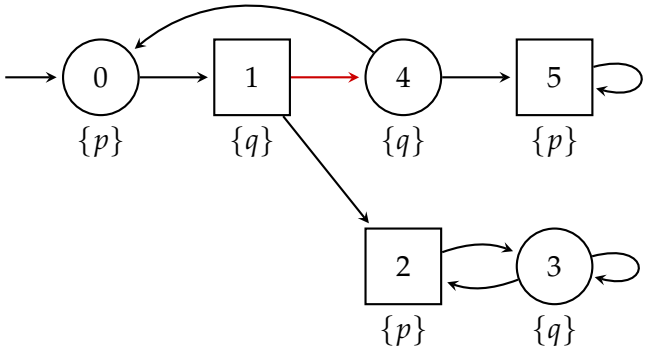

FIGURE 3.1: A motivating example for adaptive strategies

always. However, the above classical strategy for Player 0 moves it back to Vertex 0 from which p might not hold eventually 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 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 *adaptive*. Intuitively, it adapts its moves to achieve the best possible outcome after each bad move of the environment. We formalize this shortly.

3.2.2 Definition

Consider strategies σ, τ for Player 0, Player 1 respectively and let $\mathrm{p} = v_0 v_1 \dots v_n$ be a play prefix. Let the play $\rho^{\mathrm{p}}_{\sigma\tau} = \mathrm{p} v_{n+1} v_{n+2} \dots$ be the σ, τ-continuation of p, given by

$$v_{i+1} = \begin{cases} \sigma(v_0 v_1 \dots v_i) & \text{if } v_i \in V_0 \\ \tau(v_0 v_1 \dots v_i) & \text{if } v_i \in V_1 \end{cases} \qquad \text{for all } i \geq n.$$

Let $V^{\mathrm{p}}(\sigma, \tau)$ denotes the value $V(\rho^{\mathrm{p}}_{\sigma\tau})$.

We say that a strategy σ for Player 0 *enforces* truth value b from a play prefix p, if we have $V^{\mathrm{p}}(\sigma, \tau) \succeq b$ for every strategy τ for Player 1. Similarly, we say a strategy τ for Player 1 enforces truth value b from a play prefix p, if we have $V^{\mathrm{p}}(\sigma, \tau) \preceq b$ for every strategy σ for Player 0.

We also say Player i can enforce a value b from some prefix if he has a strategy that enforces the value b from that prefix.

For example, consider the game given in Figure 3.1 with the rLTL specification $\boxdot p$. Using the analysis given in Section 3.2.1, we can see that Player 0 can enforce 0011 and 0001 from the 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. Therefore, we are interested in a strategy that enforces the best possible value from each play prefix. Formally, the following is the type of strategy we are interested in.

Definition 3.2 (Adaptive Strategies)**.** In an rLTL game, a strategy σ_{opt} for Player 0 is *adaptive* if from any play prefix p, no strategy for Player 0 enforces a better truth

value than σ_{opt}, that is, if some strategy σ for Player 0 enforces a truth value of b from p, then σ_{opt} also enforces the value b from p.

A similar notion can be defined for Player 1.

3.2.3 Computation

We are interested in the following problem.

Problem 3.2. Given an rLTL game $\mathfrak{G}$, synthesize adaptive strategies for each player.

The notion of Parity games is a vital part of our algorithm to solve Problem 3.2. So let us first briefly summarize the definitions and useful properties of Parity automata and Parity games.

Parity Automata

Parity automata are similar to Büchi automata with a different accepting condition. The following is the formal definition of the deterministic version of Parity automata.

Definition 3.3 (Parity Automata)**.** A *deterministic Parity automaton* is a tuple $\mathcal{C} = (Q, \Sigma, q_0, \delta, \Omega)$ consisting of a finite, non-empty set Q of states, a finite alphabet Σ, an initial state $q_0 \in Q$, a transition function $\delta : Q \times \Sigma \to Q$, and a priority function $\Omega : Q \to \{0, 1, \ldots, d\}$ mapping each state to an integer priority, which defines the acceptance conditions.

The *run* of the deterministic Parity automaton $\mathcal{C}$ on a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ is an infinite sequence of states $\rho = q_0 q_1 \ldots \in Q^\omega$ starting at q_0 and satisfying $\delta(q_i, w_i) = q_{i+1}$ for all $i \in \mathbb{N}$. A run ρ is *accepting* if the highest priority that occurs infinitely often in the states of ρ is even. The language of a Parity automaton $\mathcal{C}$, denoted by $L(\mathcal{C})$, is the set of all infinite words for which an accepting run of $\mathcal{C}$ exists.

The main result on Parity automata that we use in our algorithm is as follows.

Theorem 3.1 (Varghese, 2014)**.** *Given a generalised (non-deterministic) Büchi automaton $\mathcal{B}$ with n states and k accepting sets, a deterministic Parity automaton $\mathcal{C}$ with $O(n!(n-1)!k^n)$ states and $2n$ priorities can be constructed such that $L(\mathcal{C}) = L(\mathcal{B})$.*

Parity Games

Parity games were introduced by Emerson, 1985 to solve the μ-calculus model checking problem. The following is the formal definition of such games.

Definition 3.4 (Parity Games)**.** A *Parity game* is a game played over a game graph $\mathcal{G} = (V, E, \Omega)$, where the finite set V of vertices is partitioned into V_0 and V_1, which are Player 0's and Player 1's vertices, respectively; $E \subseteq V \times V$ is directed edge relation; and priority function $\Omega : V \to \{0, 1, \ldots, d\}$ assigns each vertex to an integer priority.

Similar to rLTL games, a *play* $\rho = v_0 v_1 \dots$ in a Parity game is constructed by the two players, and a play ρ is *winning* for Player 0 if the highest priority that occurs infinitely often in the vertices of ρ is even. Such winning conditions are called *Parity conditions*.

Strategies are defined as it is for rLTL games. Given an initial vertex v_0, a strategy σ is *winning* for a player if all σ-plays starting from v_0 are winning for that player, and the winning region for a player is the set of vertices from which he has a winning strategy. A strategy σ is *positional* if for any two play prefix p_1 and p_2 ending in same vertex, $\sigma(\mathrm{p}_1) = \sigma(\mathrm{p}_2)$. One of the main properties of Parity games is memoryless determinacy which is formalized as follows.

Theorem 3.2 (Emerson and Jutla, 1991)**.** *For any Parity game $\mathfrak{G}$, there exists positional strategies σ_0 and σ_1 for Player 0 and Player 1, respectively, such that σ_i is a winning strategy from any vertex of the winning region of Player i.*

We are interested in solving Parity games, which is equivalent to solving the following problem.

Problem 3.3. Given a Parity game $\mathfrak{G}$, determine the winning regions and positional winning strategies for each player.

The problem of solving Parity games is in NP. However, we are only interested in a particular case of Parity games, which can be solved in polynomial time as stated in the following theorem.

Theorem 3.3 (Calude et al., 2017)**.** *A Parity game $\mathfrak{G}$ with n vertices and k priorities can be solved in $O(n^5)$ time if $k < \lg(n)$.*

Now to synthesize an adaptive strategy for Player 0 in an rLTL game $\mathfrak{G} = (\mathcal{G} = (V, E, \lambda), \varphi)$, we use the construction of generalized Büchi automaton given in Chapter 2 in the following four steps.

1. We first construct generalized (non-deterministic) Büchi automata $\mathcal{B}^b_\varphi$ such that $L(\mathcal{B}^b_\varphi) = \{w \in (2^{\mathcal{P}})^\omega : V(w, \varphi) \succeq b\}$ for all $b \in \mathbb{B}_4$.

2. We determinize each $\mathcal{B}^b_\varphi$ to get a deterministic Parity automaton $\mathcal{C}^b_\varphi = (Q^b, 2^{\mathcal{P}}, q^b_0, \delta^b, \Omega^b)$ with same language.

3. We construct five Parity games $\mathfrak{G}^b$ with the vertex set $V \times Q^b$ by taking the product of the game graph $\mathcal{G}$ and the Parity automaton $\mathcal{C}^b_\varphi$ for all $b \in \mathbb{B}_4$.

4. We apply standard techniques to solve each game $\mathfrak{G}^b$ and determine the winning set $\mathrm{win}(\mathfrak{G}^b)$ and winning strategy σ_b for Player 0.

5. We compute an adaptive strategy σ_{opt} for Player 0 as follows: for any $v_0 v_1 \dots v_k \in P^k$, we compute the corresponding paths starting from (v_0, q^b_0) of the form $(v_0, q^b_0)(v_1, q^b_1) \cdots (v_k, q^b_k)$ in each game graph of $\mathfrak{G}^b$ and determine the maximum value $b \in \mathbb{B}_4$ such that $(v_k, q^b_k) \in \mathrm{win}(\mathfrak{G}^b)$ and follow the corresponding winning strategy σ_b for the next move.

Let us now sketch these steps.

Step 1. Using the construction given in Section 2.3.3, we obtain the generalized non-deterministic Büchi automata $\mathcal{B}^b_\varphi$ such that $L(\mathcal{B}^b_\varphi) = \{w \in (2^{\mathcal{P}})^\omega : V(w, \varphi) \succeq b\}$ for all $b \in \mathbb{B}_4$. The automaton $\mathcal{B}^b_\varphi$ has $n = 5^{|\mathrm{cl}(\varphi)|} + 5$ states and at most $4|\mathrm{cl}(\varphi)|$ acceptance sets where $|\mathrm{cl}(\varphi)|$ denotes the *closure* of the formula φ.

Step 2. We determinize each $\mathcal{B}^b_\varphi$ to get a deterministic Parity automaton $\mathcal{C}^b_\varphi = (Q^b, 2^{\mathcal{P}}, q^b_0, \delta^b, \Omega^b)$ with $O(n!(n-1)!k^n)$ states and $2n$ priorities (by Theorem 3.1).

Step 3. We construct the (unlabelled) product game graph $\mathcal{G}^b = (V^b, E^b)$ of the game graph $\mathcal{G} = (V, E, \lambda)$ and the Parity automaton $\mathcal{C}^b_\varphi = (Q^b, 2^{\mathcal{P}}, q^b_0, \delta^b, \Omega^b)$ such that $V^b = V \times Q^b$ and

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

Then, we construct the function Ω^b_r that assigns the priorities to the vertices such that $\Omega^b_r(v, q) = \Omega^b(q)$. The desired Parity games are then $\mathfrak{G}^b = (\mathcal{G}^b, \Omega^b_r)$.

An induction over the length of plays in $\mathfrak{G}^b$ shows that Player 0 wins a play $\rho' = (v_0, q^b_0)(v_1, q^b_1)\ldots$ if and only if the value of the play $\rho = v_0 v_1 \ldots$ is greater than or equal to b in the game $\mathfrak{G}$.

Also given a path $\rho = v_0 v_1 \ldots v_k$ in $\mathcal{G}$, there is a unique path of the form $\rho' = (v_0, q^b_0)(v_1, q^b_1)\ldots(v_k, q^b_k)$ in the graph $\mathcal{G}^b$, that is when $q^b_{i+1} = \delta^b(q^b_i, v_i)$ for all $0 \leq i \leq k-1$ and hence, Player 0 has a winning strategy from the vertex (v_k, q^b_k) in the game $\mathfrak{G}^b$ if and only if Player 0 has a strategy that enforces the value b from the prefix $v_0 v_1 \ldots v_k$ in the game $\mathfrak{G}$.

Step 4. Then we solve these resulting Parity games $\mathfrak{G}^b$ and determine the winning set $\mathrm{win}(\mathfrak{G}^b)$ and the winning strategy σ_b for Player 0. The Parity games has $n_p = O(|V|\, n!(n-1)!k^n)$ states and $k_p = 2n$ priorities. Since $k_p < \lg(n_p)$, by Theorem 3.3, these can be solved in time $O(n_p^5) = O(|V|^5\, n!^5(n-1)!^5 k^{5n})$ and the space required is $O(n_p \log n_p \log k_p)$.

Step 5. Now we compute the adaptive strategy σ_{opt} as follows: for any play prefix $v_0 v_1 \ldots v_k \in P^k$, let $q^b_{i+1} = \delta^b(q^b_i, v_i)$ for all $i = 0, 1, \ldots, k-1$, then $(v_0, q^b_0)(v_1, q^b_1)\cdots(v_k, q^b_k)$ is the unique path in the graph $\mathcal{G}^b$ corresponding to the path $v_0 v_1 \ldots v_k$.

Then we determine the maximum value $b \in \mathbb{B}_4$ such that $(v_k, q^b_k) \in \mathrm{win}(\mathfrak{G}^b)$. Let that maximum value be b_0, then Player 0 has a strategy that enforces the value b_0 from the prefix $v_0 v_1 \ldots v_k$, and we can construct that strategy using the winning strategy σ_{b_0}. Also, for any value $b \succ b_0$, Player 0 can not win from the vertex (v_k, q^b_k) in the game $\mathfrak{G}^b$. Hence Player 0 does not have a strategy that enforces the value b from the prefix $v_0 v_1 \ldots v_k$. Therefore, Player 0 can not enforce a value more than b_0 from

the prefix $v_0v_1 \dots v_k$. Hence, we set $\sigma_{opt}(v_0v_1 \dots v_k)$ to be the first entry of the pair $\sigma_{b_0}(v_k, q_k^{b_0})$.

Similarly, we can compute an adaptive strategy for Player 1. Hence, an adaptive strategy for a player in an rLTL game can be computed in doubly-exponential time and space. Furthermore, note that adaptive strategies require memory. By Theorem 3.2, we know that winning strategies in Parity games are memoryless, and the Parity games we constructed in step 3 have doubly-exponential size. Therefore, adaptive strategies require doubly exponential memory. In total, we obtain the following result.

Theorem 3.4. *Given an* rLTL *game* $\mathfrak{G} = (\mathcal{G} = (V, E, \lambda), \varphi)$*, an adaptive strategy of a player can be computed in time* $O(|V|^5\, n!^5 (n-1)!^5 k^{5n})$*, where* $n = 5^{|cl(\varphi)|} + 5$ *and* $k = 4|cl(\varphi)|$*. Moreover, each player has an adaptive strategy with the memory size of* $O(|V|\, n!(n-1)!k^n)$*.*

3.3 Strongly Adaptive Strategies

In this section, we start by formalizing the notion of bad moves, which we need later. Then using a modified version of Figure 3.1, we describe the importance of strongly adaptive strategies and why we need them. Then we show that such a strategy may not exists for some games, whereas adaptive strategies always exist. Moreover, we give a doubly-exponential algorithm to compute a strongly adaptive strategy, if one exists at all.

3.3.1 Bad Moves

We already have used the notion of bad moves for the example given in Section 3.2, and the name itself is intuitive. Formally, we say a move of a player from some play prefix p to a vertex v is *bad* if that player can enforces some value b from the prefix p, but he can not enforce that value b from the prefix $\mathrm{p}v$.

As we have seen in the example given in Section 3.2.1, Player 1 can move the token from Vertex 1 to 2 and can enforce the value 0001 from the prefix 01 through the path 0123^ω. Suppose he moves the token from 1 to 4 (instead of 2), then Player 0 can enforce the value 0111 by repeatedly visiting Vertex 5. Hence Player 1 no longer can enforce the value 0001 from the prefix 014. Therefore, the move from the prefix 01 to Vertex 4 made by Player 1 is a bad move.

3.3.2 Motivating Example

Suppose we modified the arena given in Figure 3.1 by adding an edge from 0 to 2 as shown in the Figure 3.2. Using the analysis given in Section 3.2.1, we know that the strategy of moving the token along $\{0 \to 1, 4 \to 5, 3 \to 2\}$ is adaptive for Player 0. Another adaptive strategy for Player 0 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 Vertex 4, and hence, Player 1 can never make a bad move. However, it also means

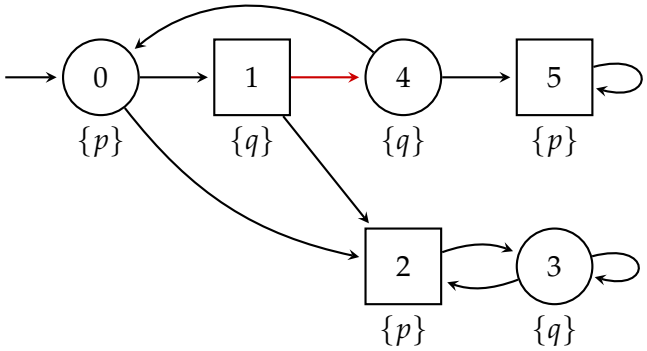

FIGURE 3.2: A motivating example for strongly adaptive strategies

that there cannot be a play where p holds eventually always; whereas 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 to 1, Player 0 might be able to enforce the value 0111 at some point; whereas using the other one, he can enforce at most 0001 from any prefix of a play. Hence, the earlier one is definitely better. Therefore, we also consider another type of strategy, which is adaptive, as well as maximizes the chance of the environment making a bad move. We call such a strategy *strongly adaptive*.

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 his opponent can make in both choices and determine the choice in which he is more likely to get better value. To capture this, we define a new notion of strategy called strongly adaptive strategy, which we will formalize shortly.

3.3.3 Definition

Consider an rLTL game. For any $0 \leq i \leq 4$, we say the strategy σ *enforces truth value* b *with* i *bad moves* from a play prefix p if the followings hold: let A_i be the set of plays $\rho = \mathrm{p}\rho'$ which are the continuation of play prefix p by strategy σ such that ρ' contains exactly i bad moves of Player 1, then A_i is non-empty and for any play $\rho \in A_i$, we have $V(\rho) \succeq b$. Note that σ enforces a value b with 0 bad move from some prefix p if and only if σ enforces the value b from the prefix p.

Now for a strategy σ of Player 0 and a play prefix p, let the *enforced values* $e(\mathrm{p}, \sigma)$ be a 5-tuple such that for each $0 \leq i \leq 4$, $e_i(\mathrm{p}, \sigma)$ is defined as follows: it is $\perp$ if σ does not enforce any value with i bad moves from the prefix p; else it is the maximum value σ enforces with i bad moves from the prefix p. We define $b \succ \perp$ for any truth value $b \in \mathbb{B}_4$.

For example, consider the game given in Section 3.3.2. Let σ be the strategy of moving the token along $\{0 \to 1, 4 \to 5, 2 \to 3\}$. Formally, σ is the strategy for Player 0 such that $\sigma(V^*0) = 1, \sigma(V^*4) = 5$ and $\sigma(V^*2) = 3$. Then as we have discussed earlier, the maximum value σ enforces from any of the prefixes $0, 01, 02$ is 0001. However, if Player 1 makes a bad move from 01 by moving the token to

4, then σ enforces 0111 from 014. Therefore, σ enforces 0111 with one bad move from 0. Hence, $e(\sigma, 0) = (0001, 0111, \bot, \bot, \bot)$ (since no more bad move of Player 1 is possible). However, for any adaptive strategy σ' that moves the token from the prefix 0 to 2, we have $e(\sigma', 0) = (0001, \bot, \bot, \bot, \bot)$ as Player 1 can never make a bad move starting from 0. Therefore, σ is a better strategy than any adaptive strategy that moves along $0 \to 2$.

Intuitively, we are interested in an adaptive strategy that gives more opportunity to the opponent to make bad moves that ensure a better value. Moreover, to choose between two adaptive strategies, we first compare the value they enforce with one bad move from any prefix; if that is equal for all prefixes, then we compare the value they enforce with two bad moves, and so on. Formally, generalizing the Definition 3.2, we are interested in the following type of strategy:

Definition 3.5. In an rLTL game, a strategy σ_0 for Player 0 is *strongly adaptive* if for any play prefix $\mathtt{p}$, no adaptive strategy for Player 0 has a better enforced values than σ_0, that is, for every adaptive strategy σ for Player 0, it holds that

$$e(\mathtt{p}, \sigma_0) \succeq_{lex} e(\mathtt{p}, \sigma).$$

Note that any strongly adaptive strategy is also adaptive by definition.

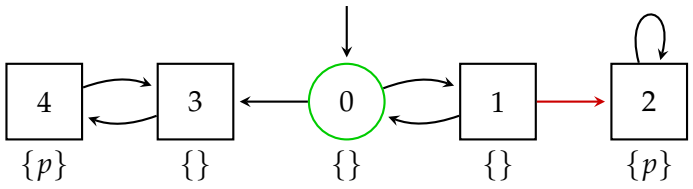

FIGURE 3.3: An rLTL game with no strongly adaptive strategy

3.3.4 Existence of Strongly Adaptive Strategies

We know that a strongly adaptive strategy is the best one among all adaptive strategies. However, there is a catch in the definition, that is, such strategy may not even exist for some games; whereas an adaptive strategy always exists. For example, consider the arena given in Figure 3.3 with initial vertex 0. It is clear that Player 0 can enforce the value 0011 from any prefix $(0+1)^*$ by eventually moving to Vertex 3. And if at some point, Player 1 makes a bad move of $1 \to 2$, then Player 0 can enforce 0111 by staying at Vertex 2 forever. Hence, any adaptive strategy for Player 0 eventually visits Vertex 3 unless Player 1 makes a bad move. Now for any $i \geq 0$, let A_i be the set of all adaptive strategy σ for Player 0 such that $\sigma((01)^i 0) = 3$ and $\sigma((01)^k 0) = 1$ for all $k < i$. Then clearly $\cup_{i \geq 0} A_i$ is the set of all adaptive strategies for Player 0 and for any $\sigma_i \in A_i$, $\sigma_{i+1} \in A_{i+1}$, we have

$$e((01)^i, \sigma_{i+1}) = (0011, 0111, \bot, \bot, \bot) \succ_{lex} (0011, \bot, \bot, \bot, \bot) = e((01)^i, \sigma_i).$$

Hence, the strategy σ_{i+1} is better than σ_i. This also seems correct since Player 1 has more opportunities to make a bad move (of moving the token from 1 to 2) if Vertex 1 is visited more times. Then for every adaptive strategy, there is a better one; hence there does not exist a best one.
So now we are interested in the following problem.

Problem 3.4. Given an rLTL game $\mathfrak{G}$, synthesize, if possible, a strongly adaptive strategy for Player 0; or determine that no such strategy exists.

3.3.5 Computation

We solve the Problem 3.4 for an rLTL game by constructing an extended arena, where positional adaptive strategy always exists. Then we show that it is equivalent to solve the problem for that extended arena.

Let $\mathfrak{G} = (\mathcal{G} = (V, E, \lambda), \varphi)$ be an rLTL game. Let $\mathcal{C}^b_\varphi = (Q^b, 2^{\mathcal{P}}, q^b_0, \delta^b, \Omega^b)$ be the deterministic Parity automata such that $L(\mathcal{C}^b_\varphi) = \{w \in (2^{\mathcal{P}})^\omega : V(w, \varphi) \succeq b\}$, and $\mathfrak{G}^b$ be the Parity game (product of the game graph $\mathcal{G}$ and the Parity automaton $\mathcal{C}^b_\varphi$) computed using the steps of the algorithms given in Section 3.2.3. Let $\text{win}(\mathfrak{G}^b)$ be the winning set for Player 0 in the game $\mathfrak{G}^b$.

Consider an extended rLTL game $\mathfrak{G}' = (\mathcal{G}', \varphi)$, where $\mathcal{G}'$ is a game graph with vertex set $V \times Q^{0000} \times Q^{0001} \times \cdots \times Q^{1111}$ constructed by taking product of the game graph $\mathcal{G}$ and all the Parity automata $\mathcal{C}^b_\varphi$ such that

$$\Big((v_1, q_1^{0000}, q_1^{0001}, q_1^{0011}, q_1^{0111}, q_1^{1111}), (v_2, q_2^{0000}, q_2^{0001}, q_2^{0011}, q_2^{0111}, q_2^{1111})\Big) \in E(\mathcal{G}')$$
$$\Leftrightarrow (v_1, v_2) \in E \text{ and } \delta^b(q_1^b, \lambda(v)) = q_2^b \text{ for all } b \in \mathbb{B};$$

and $\lambda'(v, q^{0000}, \ldots, q^{1111}) = \lambda(v)$ for all $v \in V, q^b \in Q^b$. It is easy to see that there is an one to one correspondence between paths in both games $\mathfrak{G}$ and $\mathfrak{G}'$. Since the rLTL specification and the labels are similar in both games, there is also a one to one correspondence between strategies (adaptive strategies, strongly adaptive strategies) in both games $\mathfrak{G}$ and $\mathfrak{G}'$. So solving Problem 3.4 for the game $\mathfrak{G}$ is equivalent to solving it for the game $\mathfrak{G}'$.

Now using the analysis of the algorithm given in Section 3.2.3, we have the followings in the rLTL game $\mathfrak{G}'$:

- Player 0 can enforce a value b from a prefix ending in $(v, q^{0000}, \ldots, q^{1111})$ if and only if $(v, q^b) \in \text{win}(\mathfrak{G}^b)$. Since this notion only depends on the end vertex, we say Player 0 can enforce a value b from the vertex $\bar{v} = (v, q^{0000}, \ldots, q^{1111})$ if he can enforce b from each prefix ending in $\bar{v}$.
- $\text{win}(b) = \{(v, q^{0000}, \ldots, q^{1111}) \in V(\mathcal{G}') \mid (v, q^b) \in \text{win}(\mathfrak{G}_b)\}$ is the set of vertices from which Player 0 can enforce the value b.

- For any truth value $b \prec 1111$, let $b+1$ be the smallest value bigger than b. Then $W(b) = \text{win}(b) \setminus \text{win}(b+1)$ is the set of vertices from which the maximum value Player 0 can enforce is b.
- Every bad moves of Player 1 correspond to a (unique) edge (u, v) such that $u \notin \text{win}(b)$ but $v \in \text{win}(b)$ for some value b. Let call such edges *bad*. Then the set of bad edges is the disjoint union of the following sets over all values $b \prec b'$:

$$\text{bad}(b, b') = \{(u, v) \in E(\mathcal{G}') \mid u \in W(b), v \in W(b')\}$$

(Note that for $b \prec b'$, $(u, v) \in \text{bad}(b, b')$ only when $u \notin \text{win}(b')$ but $v \in \text{win}(b')$)

The next lemma follows from memoryless determinacy of Parity games.

Lemma 3.1. *A strategy σ_0 for Player 0 in the game $\mathfrak{G}'$ is strongly adaptive if and only if for every play prefix $\mathtt{p}$ ending in v, it holds that*

$$e(\mathtt{p}, \sigma_0) = \max\{e(v, \sigma) \mid \sigma \textit{ is an adaptive strategy for Player } 0 \textit{ in the game } \mathfrak{G}'\}.$$

Proof. For any play prefix $\mathtt{p}$, let $e_m(\mathtt{p})$ denotes the maximum of $e(\sigma, \mathtt{p})$ over all adaptive strategies σ. Then by definition of strongly adaptive strategies, it is enough to show that for any two prefixes $\mathtt{p}_1$ and $\mathtt{p}_2$ ending in the same vertex, it holds that $e_m(\mathtt{p}_1) = e_m(\mathtt{p}_2)$. Suppose $e_m(\mathtt{p}_1) < e_m(\mathtt{p}_2)$. Let σ_i be an adaptive strategy that maximizes the enforced values of $\mathtt{p}_i$ over all the adaptive strategies. Then consider another strategy σ for Player 0 such that for any play prefix $\mathtt{p}_1\mathtt{p}$, it holds that $\sigma(\mathtt{p}_1\mathtt{p}) = \sigma_2(\mathtt{p}_2\mathtt{p})$ and for any other prefixes it coincides with σ_1. It is clear that $e(\sigma, \mathtt{p}) = e(\sigma_2, \mathtt{p}_2)$. Since σ never make a bad move and follows an adaptive strategy eventually, it is an adaptive strategy that satisfies $e(\sigma, \mathtt{p}) > e(\sigma_1, \mathtt{p})$, which contradicts the maximality of $\sigma_{\mathtt{p}}$. □

Now we can solve Problem 3.4 for Player 0 in the rLTL game $\mathfrak{G}'$ using the following steps.

1. For each possible enforced value t, we first recursively compute the set of vertices $V[t]$ containing all vertices v such that t is the maximum enforced values of v over all adaptive strategies.
2. We then recursively compute if one exists a strategy σ_t for Player 0 in each subgraph $\mathcal{G}$ restricted to $V[t]$, which satisfies the followings:
   (a) Parity winning condition of $\mathcal{G}^{t_0}$ (where $t = (t_0, t_1, t_2, t_3, t_4)$);
   (b) If $t_1 \neq \bot$, then let t' be the tuple $(t_1, t_2, t_3, t_4, \bot)$. Then from any play prefix, there exists a σ_t-play that visits a vertex v from which there is an edge to $V[t']$ in $\mathcal{G}'$.
   (c) For any enforced value $t'' \prec_{lex} t'$ with $t''_i \prec t'_i$ for some i, no σ_t-play visits a vertex v from which there is an edge to $V[t'']$ in $\mathcal{G}'$.

3. If σ_t exists for all possible enforced values t, then combine all strategies σ_t to get a strongly adaptive strategy σ for $\mathfrak{G}'$; else, there is no strongly adaptive strategy for $\mathfrak{G}'$.

For step 2 of the algorithm, we use the reduction to another type of games, called *Obliging games*. So, before describing the details of the algorithm, let us recapitulate the definitions and useful results on Obliging games.

Obliging Games

An *Obliging game* is a two-player game introduced by Chatterjee, Horn, and Löding, 2010. It is played on an arena $\mathcal{G}$ with two winning conditions S and W, called strong and weak, respectively. The objective of Player 0 is to ensure the strong winning condition while allowing Player 1 to cooperate with him to fulfill the weak winning condition additionally. Such winning strategies are therefore called *gracious*. We are only interested in the Obliging games where the strong condition is a Parity condition, and the weak one is a Büchi condition. Such games are called Parity/Büchi Obliging games and a gracious strategy for such games are formalized below.

Problem 3.5. Given a Parity/Büchi Obliging game on a game graph $\mathcal{G}$ with Parity condition S and Büchi condition W, synthesize a strategy σ for Player 0 that satisfies the followings:

- every σ-play is winning w.r.t. S,
- for every finite σ-play, there is a continuation by σ which is winning w.r.t. W;

or determine that no such strategy exist.

The following theorem characterizes the solution of Problem 3.5 for a Parity/Büchi Obliging game by a reduction to a Parity game using the results by Chatterjee, Horn, and Löding, 2010. As Parity games are decidable and gracious strategy in such games can be effectively computed.

Theorem 3.5. *A Parity/Büchi Obliging game with n states, a Parity condition with $2k$ priorities, and a Büchi condition can be reduced into a Parity game with $O(n)$ states and $2k+2$ priorities. Moreover, if Player 0 has a gracious strategy in such Obliging games, then he has a gracious strategy with the memory of size at most $4k$.*

Let us now sketch the steps of the algorithm.

Step 1. For any vertex v, let $e_m(v)$ be the maximum enforced values of v over all adaptive strategies as in the proof of Lemma 3.1. For a tuple t of truth values, let $V[t]$ be the set of vertices v such that $e_m(v) = t$. Let us fix an adaptive strategy σ_0 for Player 0 in the game $\mathfrak{G}'$. Now we can recursively compute $V[t]$ for each possible t as follows. First of all, it is clear that

$$W(b) = \bigcup_{t=(b,\ldots)} V[t].$$

- For $t = (1111, \perp, \perp, \perp, \perp)$, it is easy to see that $V[t] = W(1111)$.
- For $t = (0111, 1111, \perp, \perp, \perp)$, we claim that $V[t]$ is the set of vertices in $W(0111)$ from which the set $W(1111)$ is reachable. If $e_m(v) = t$ for some vertex v, then $W(1111)$ is reachable from v. Conversely, let $W(1111)$ is reachable from v by a path P. Then consider a strategy σ which assigns each prefix p of P ending in V_0 to the vertex that occurs after p in the path P; and for other prefixes, it coincides with σ_0. Since the strategy σ eventually follows an adaptive strategy and never makes a bad move, it is also adaptive. Clearly, from vertex v, there exists a σ-play containing a bad move by Player 1. Since $v \in W(0111)$, the maximum possible value for $e_m(v)$ is t. Hence, $e_m(v) = t$.
- For $t = (0111, \perp, \perp, \perp, \perp)$, it is easy to see that

$$V[t] = W(0111) \setminus V[(0111, 1111, \perp, \perp, \perp)].$$

- For $t = (0011, 1111, \perp, \perp, \perp)$, a vertex $v \in W(0011)$ belongs to $v[t]$ iff there exists an adaptive strategy σ such that in $\mathcal{G}'_\sigma$, a path from v to $W(1111)$ exists but no path from v to $W(0111)$ exist. Hence, $V[t] \subseteq W(0011) \setminus \text{Reach}_1(W(0111))$, where $\text{Reach}_1(F)$ denotes the set of vertices in $V(\mathcal{G}')$ from which Player 1 can force the token to reach F. For any vertex set F, $\text{Reach}_1(W(0111))$ is the union of the sequence of sets $\left(\text{Attr}_i^1(F)\right)_{i\geq 0}$ with the property that from any position of $\text{Attr}_i^1(F)$, Player 1 can enforce to reach F in at most i many steps and these can be computed inductively as follows:

$$\begin{aligned} \text{Attr}_0^1(F) &= F \\ \text{Attr}_{i+1}^1(F) &= \text{Attr}_i^1(F) \bigcup \{v \in V_0(\mathcal{G}') : N(v) \subseteq \text{Attr}_i^1(F)\} \\ &\quad \bigcup \{v \in V_1(\mathcal{G}') : N(v) \cap \text{Attr}_i^1(F) \neq \emptyset\} \end{aligned}$$

  Now we consider the subgraph $\mathcal{G}_t$ which is $\mathcal{G}'$ restricted to $W(0011) \setminus \text{Reach}_1(W(0111))$. Let $\text{win}(\mathcal{G}_t)$ be the winning set for the Parity game with arena $\mathcal{G}_t$ and Parity condition same as in the game $\mathcal{G}^{0011}$ and let $\sigma(\mathcal{G}_t)$ be the winning strategy for Player 0.

  As we have seen in the second case, $V[t]$ is the set of vertices from which $\text{bad}(0011, 1111)$ reachable in $\mathcal{G}_t$, since, for such vertices, the strategy that maximizes the enforced values is the one that follows the corresponding path and for other prefixes, it coincides with $\sigma(\mathcal{G}_t)$.
- For $t = (0011, 0111, 1111, \perp, \perp)$, we can see that $V[t]$ is the set of vertices in $W(0011) \setminus V[(0011, 1111, \perp, \perp, \perp)]$ from which $V[(0111, 1111, \perp, \perp, \perp)]$ is reachable.

- Similarly, for $t = (0011, 0111, \bot, \bot, \bot)$, $V[t]$ is the set of vertices in $W(0011) \setminus (V[(0011, 1111, \bot, \bot, \bot)] \cup V[(0011, 0111, 1111, \bot, \bot)]$ from which $W(0111)$ is reachable and $V[(0011, \bot, \bot, \bot, \bot)]$ is the rest of vertices in $W(0011)$.
- Similarly, $V[t]$ can be computed for other possible values of t.

Step 2. Now we need to check whether there exists a strategy σ such that $e(\mathrm{p}, \sigma) = e_m(v)$ for any prefix p ending in v. It is clear from the construction that there is no edge from a Player 0's vertex in $V[t]$ to $V[t']$ if $t \prec_{lex} t'$. Moreover, a strategy satisfying the above property would never use an edge from $V[t']$ to $V[t]$ if $t \prec_{lex} t'$. Hence, starting from a vertex in $V[t]$, unless Player 1 makes a bad move, the token stays in $V[t]$.

We first show that any strategy σ satisfying properties given in 2a,2b, 2c (when restricted to $V[t]$) is indeed a strongly adaptive strategy. Property 2a ensure that the strategy is adaptive. Moreover, the other two properties ensures that σ enforces t_i (if $t_i \neq \bot$) with i bad moves. So we recursively check (and compute if it exists) a strategy for the game restricted to the vertex set $V[t]$ which satisfies the required properties.

- For $t = (1111, \bot, \bot, \bot, \bot)$, we can just use the adaptive strategy σ_0 for the prefixes in $V[t]$.
- For $t = (0111, 1111, \bot, \bot, \bot)$, we need to check if there exists an adaptive strategy σ such that for all finite σ-play in $V[t]$, there exists a continuation by σ in $V[t]$ which visits a vertex v from which there is an edge to $W(1111)$.

  Note that if σ satisfies the property of visiting a vertex set F from every finite σ-play then once it reaches F from some prefix, using the same property it visits F again. Hence, it visits the vertex set F infinitely often from every finite σ-play. The converse is also trivially true. Therefore, it is enough to check if there exists a uniformly gracious strategy for Player 0 in the Parity/Büchi Obliging game with arena $\mathcal{G}'$ restricted to vertex set $V[t]$, Parity condition of $\mathcal{G}^{0111}$ and Büchi condition $\{u \in V[t] : \exists v \text{ s.t. } (u, v) \in \mathrm{bad}(0111, 1111)\}$.
- For $t = (0111, \bot, \bot, \bot, \bot)$, we can just use the adaptive strategy σ_0 for the prefixes in $V[t]$.
- For $t = (0011, 1111, \bot, \bot, \bot)$, it is easy to see that there does not exists an edge from $V[t]$ to $W(0111)$. So, we need to check if there exists a uniformly gracious strategy for Player 0 in the Parity/Büchi Obliging game with arena $\mathcal{G}'$ restricted to vertex set $V[t]$, Parity condition of $\mathcal{G}^{0011}$ and Büchi condition $\{u \in V[t] : \exists v \text{ s.t. } (u, v) \in \mathrm{bad}(0011, 1111)\}$.
- Similarly, for $t = (0011, 0111, 1111, \bot, \bot)$, we need with $\mathrm{bad}(0111, 1111)$ infinitely often. Therefore, it is enough to check if there exists a uniformly gracious strategy for Player 0 in the Parity/Büchi Obliging game with arena $\mathcal{G}'$ restricted to vertex set $V[t]$, Parity condition of $\mathcal{G}^{0011}$ and Büchi condition $\{u \in V[t] : \exists v \in V[(0111, 1111, \bot, \bot, \bot)] \text{ s.t. } (u, v) \in \mathrm{bad}(0111, 0111)\}$.

- Similarly, for all other tuples t, we can reduce the problem to an Obliging game, or we can use the strategy σ_0.

Since the game graph $\mathcal{G}'$ has $O(|V|\, n!^5 (n-1)!^5 k^{5n})$ states and the parity conditions $\mathcal{G}^b$ requires $2n$ priorities, using Theorem 3.5, above Parity/Büchi Obliging games can be reduced to Parity games with $N = O(|V|\, n!^5 (n-1)!^5 k^{5n})$ states and $K = 2n+2$ priorities. Such games can then be solved in $O(N^5)$ time and $O(N \log N \log K)$ space.

Step 3. So, if such strategies exist for each $V[t]$, then we can combine them into a strongly adaptive strategy for the game $\mathfrak{G}'$, which can again be reduced to a strongly adaptive strategy for the original game $\mathfrak{G}$. Else there is no strongly adaptive strategy for the game $\mathfrak{G}$.

Strongly adaptive strategies also require doubly exponential memory, since the Obliging games (and Parity games) constructed in the algorithms have doubly exponential size, and by Theorem 3.5, gracious strategies in such Obliging games require memory of size at most $4n$. In total, we get the following result.

Theorem 3.6. *Problem 3.4 for an* rLTL *game* $\mathfrak{G} = (\mathcal{G} = (V, E, \lambda), \varphi)$ *can be solved in time* $O(|V|^5\, n!^{25} (n-1)!^{25} k^{25n})$, *where* $n = 5^{|cl(\varphi)|} + 5$ *and* $k = 4|cl(\varphi)|$. *Moreover, if Player* 0 *has a strongly adaptive strategy in such* rLTL *games, then he has a strongly adaptive strategy with the memory size of* $O(|V|\, n!^6 (n-1)!^4 k^{5n})$.

Therefore, both adaptive and strongly adaptive strategies can be synthesized in doubly-exponential time and space. 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.

Chapter 4

Conclusion

In this thesis, we have considered the problem of synthesizing the most robust controller with respect to a criterion based on Robust LTL. In Chapter 2, we have discussed the syntax and semantics of rLTL and described how robust its formulas are. We also saw that LTL and rLTL are equally expressive.

In Chapter 3, we have focused on our main problem. We have defined a modified version of rLTL games. With an example, we have shown why the strategies computed by current algorithms are not optimal and why adaptive strategies are better. Then we have given an algorithm to compute an adaptive strategy for any player in an rLTL game using reduction to Parity games.

We have also presented an example of a strongly adaptive strategy and described how it maximizes the chances of the opponent making a bad move. However, we have given an example showing that such a strategy may not exist. Finally, using a reduction to a series of Parity games and Obliging games, we have given an algorithm that decides its existence and computes it if one exists at all.

We have shown that both the algorithm takes doubly-exponential time in the length of the given rLTL formula, which is same as the running time of the algorithms for classical synthesis problem of LTL and rLTL. Hence, computing both the adaptive and strongly adaptive strategies is not harder than the classical one.

Bibliography

Baier, Christel and Joost-Pieter Katoen (2008). *Principles of Model Checking (Representation and Mind Series)*. The MIT Press. ISBN: 026202649X.

Calude, Cristian S. et al. (2017). "Deciding Parity Games in Quasipolynomial Time". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: Association for Computing Machinery, 252–263. ISBN: 9781450345286. DOI: 10.1145/3055399.3055409. URL: https://doi.org/10.1145/3055399.3055409.

Chatterjee, Krishnendu, Florian Horn, and Christof Löding (2010). "Obliging Games". In: *CONCUR 2010 - Concurrency Theory*. Ed. by Paul Gastin and François Laroussinie. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 284–296. ISBN: 978-3-642-15375-4.

Emerson, E. A. and C. S. Jutla (1991). "Tree automata, mu-calculus and determinacy". In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pp. 368–377. DOI: 10.1109/SFCS.1991.185392.

Emerson, E. Allen (1985). "Automata, tableaux, and temporal logics". In: *Logics of Programs*. Ed. by Rohit Parikh. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 79–88. ISBN: 978-3-540-39527-0.

Priest, Graham (2009). "Dualising intuitionictic negation". In: *Principia: an international journal of epistemology* 13.2, pp. 165–184.

Tabuada, Paulo and Daniel Neider (2016). "Robust Linear Temporal Logic". In: *CSL*. Vol. 62. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 10:1–10:21.

Varghese, Praveen Thomas Methrayil (2014). "Parity and generalised Büchi automata : determinisation and complementation". PhD thesis. University of Liverpool, UK. URL: http://repository.liv.ac.uk/2027479/.