Title
A mithrilian approach to safety and robustness of autonomous cyber-physical systems

Author
Anevlavis, Tzanis

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

A *mithrilian* approach to safety and robustness of autonomous cyber-physical systems

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Tzanis Anevlavis

2022

ABSTRACT OF THE DISSERTATION

A *mithrilian* approach to safety and robustness of autonomous cyber-physical systems

by

Tzanis Anevlavis
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2022
Professor Paulo Tabuada, Chair

Every engineer dreams of and strives for the day that more and more aspects of the daily life become smart, efficient, and automated. Smart grids, smart cities, mobility on demand and autonomous vehicles, even medical devices are a few domains with already significant progress towards a fully autonomous future. In such an increasingly autonomous world, guaranteeing safety and correctness of design and implementation of Cyber-Physical Systems (CPSs) is constantly under the spotlight. There is no room for error when a system interacts with and affects human lives. Consequently, we want to design systems that are safe and correct even when interacting with unknown, changing environments. We want the algorithms designing those systems to be efficient upon execution and provide formal guarantees about safety and correctness. In other words, they should act as a *lightweight and impenetrable armor* for the system.

Such armors are found in the work of the father of modern fiction literature, J.R.R.Tolkien. He conceived in his work the metal *mithril* [THO01], and mithril armors are described as "light and yet harder than tempered steel". Inspired by this, the approaches presented in this dissertation are termed *mithrilian* as they are characterized by computational efficiency

and provide formal safety and robustness guarantees. More specifically, this dissertation discusses formal methods[1] in control systems and is separated in two parts:

1. The first part concerns the *synthesis of safety controllers* via *Robust Controlled Invariant Sets* (RCISs). A safety enforcing controller keeps the state of the system within a set of safe states notwithstanding the presence of uncertainties. Using RCISs to design safety controllers is natural, as any trajectory starting within these sets, can always be forced to remain therein. On these grounds, we revisit the problem of computing RCISs for discrete-time linear systems. Departing from previous approaches, we consider implicit, rather than explicit, representations for controlled invariant sets. Moreover, by considering such representations in the space of states and finite input sequences we obtain *closed-form* expressions for controlled invariant sets. An immediate advantage is the ability to handle high-dimensional systems since the closed-form expression is computed in a single step rather than iteratively. We present thorough case studies illustrating that in safety-critical scenarios the implicit representation suffices in place of the explicit RCIS. The proposed method is complete in the absence of disturbances and we provide a weak completeness result when disturbances are present.

2. In the second part, we switch gears and consider the problem of *guaranteeing robustness* in system design. Robustness is introduced in system verification by *robust Linear Temporal Logic* (rLTL). As CPSs inevitably become increasingly complex, the ability to completely guarantee correctness of their design and implementation via exhaustive testing fades. Most work in formal methods has focused on system correctness, i.e., in ensuring that systems are guaranteed to meet their design specifications. We argue that correctness is necessary, but not sufficient for a good design when a reactive system interacts with an ever-changing uncontrolled environment.Thus, in addition to correctness, systems should also be designed to be robust, i.e., small deviations from the assumptions made at design time should lead to,

[1]We aim to *Motivate Invariance THeory and Robustness In Logic* (MITHRIL) for control, design, and analysis of Cyber-Physical Systems.

at most, small violations of the design specifications. The contribution here lies in refining the current complexity upper bound of rLTL verification and developing a tool for rLTL verification. More specifically, we identify a large fragment of rLTL for which the verification problem can be efficiently solved, i.e., verification can be done by using an automaton, recognizing the behaviors described by the rLTL formula φ, of size at most $\mathcal{O}\left(3^{|\varphi|}\right)$, where $|\varphi|$ is the length of φ. This result improves upon the previously known bound of $\mathcal{O}\left(5^{|\varphi|}\right)$ and is closer to the LTL bound of $\mathcal{O}\left(2^{|\varphi|}\right)$. The usefulness of this fragment is demonstrated by a number of case studies showing its practical significance in terms of expressiveness, the ability to describe robustness, and the fine-grained information that rLTL brings to the process of system verification. Moreover, these advantages come at a low computational overhead with respect to LTL verification. To perform rLTL verification, we developed *Evrostos*, the first tool for model-checking rLTL formulas in this fragment.

The dissertation of Tzanis Anevlavis is approved.

Jyotirmoy V. Deshmukh

Christina Fragouli

Lieven Vandenberghe

Paulo Tabuada, Committee Chair

University of California, Los Angeles

2022

To my family ...

Αυτό είναι για εσάς.

TABLE OF CONTENTS

I Robust controlled invariant sets: implicit closed-form representations and applications **1**

1 Introduction **2**

1.1 Related literature 3

2 Problem formulation **6**

3 Implicit representation of controlled invariant sets for linear systems . . **10**

3.1 General implicit robust controlled invariant sets 11

3.2 Finite reachability constraints 12

3.3 Implicit robust controlled invariant sets in closed-form 13

3.4 Obtaining an explicit robust controlled invariant set 16

4 A hierarchy of controlled invariant sets **18**

5 Performance bound for the proposed method **22**

6 Implicit invariant sets in practice: controlled invariant sets in one move **29**

6.1 Supervision of a nominal controller 29

6.2 Safe online planning 31

6.3 Safe hyper-boxes 32

7 Case studies **36**

7.1 cis2m: a library for computing controlled invariant sets in 2 moves and closed-form implicit representations . . . 36
7.2 Supervision of a quadrotor for obstacle avoidance using explicit RCIS . . . 36
7.3 Safe online planning using implicit RCIS . . . 40
7.4 Scalability and quality . . . 44
7.4.1 Scalability of implicit invariant sets . . . 44
7.4.2 Quality of the computed sets and comparison to other methods . . . 46

8 Discussion & Conclusion . . . 50

II rLTL verification: now faster than ever before! 52

9 Introduction . . . 53
9.1 In a labyrinth of robustness . . . 55
9.2 Beyond rLTL verification . . . 59

10 Review of Linear Temporal Logic (LTL) and LTL model-checking . . . 61

11 The Syntax and Semantics of Robust Linear Temporal Logic . . . 67
11.1 rLTL Syntax . . . 67
11.2 The $\text{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ fragment . . . 68
11.2.1 Why 5 truth values? . . . 69
11.2.2 da Costa Algebras . . . 70
11.2.3 Semantics of $\text{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ on da Costa Algebras . . . 74
11.3 Full $\text{rLTL}(\mathcal{P})$ . . . 77
11.3.1 Why 5 truth values? . . . 77

11.3.2 Semantics of full rLTL($\mathcal{P}$) 78
11.4 Review of rLTL semantics 80
11.5 Examples 81
11.5.1 The usefulness of implications that are **not true** 81
11.5.2 GR(1) in rLTL 81
11.5.3 Non-counting formulae 82
11.5.4 Counting with "robust release" 83
11.5.5 Non-decomposition of "robust until" 83

12 The rLTL model checking problem 85
12.1 Relating LTL and rLTL 85
12.2 rLTL model-checking: definitions and background 88
12.3 Improved bounds for rLTL model-checking 89

13 A fragment for efficient rLTL model-checking 91
13.1 Main results 92
13.2 Bit-wise independence of rLTL$_{\setminus\{\Rightarrow\}}(\mathcal{P})$ and insights on the robust implication 94
13.3 Introducing temporal testers 98
13.4 Refined complexity bounds 102

14 Case studies 105
14.1 Evrostos: the rLTL verifier 105
14.2 Case study 1: Aircraft Wheel Brake System (WBS) 106
14.2.1 Formal specifications: 108
14.2.2 Example scenario: 109

14.3 Case study 2: Meaningful reactivity rLTL patterns 110
14.3.1 Benchmark: Rigorous Examination of Reactive Systems (RERS) . . . 111
14.4 Case study 3: Studying the complexity blowup between LTL and rLTL . . . 112
14.5 Case study 4: Scalability 114

15 Conclusion 116

A Appendix 117
A.1 Proof of Lemma 12.1.1 117
A.2 Proof of Lemma 13.4.1 119

References 124

LIST OF FIGURES

4.1 RCIS corresponding to $L = 1$ (white), $L = 2$ (gray), $L = 3$ (teal), $L = 4$ (green), $L = 5$ (yellow), and $L = 6$ (orange) for a double integrator. Safe set in blue. . . 21

6.1 The overall safe online planning framework. 31

7.1 Quadrotor operational region. Obstacles in purple transparent boxes. Nominal trajectory (red), corrected trajectory (blue), supervision active (green). 38

7.2 Quadrotor trajectory in x-y plane: nominal trajectory (red), corrected trajectory (blue), supervision active (green). 39

7.3 Robot operational space: initial position (yellow arrowhead), target position (cyan), unsafe region (dark area). 41

7.4 Simulation at $t = 0s$ (top), $40s$ (middle), and 78.6s (bottom). Left: reference path (red), actual trajectory (blue), LiDAR measurements (blue disks). Right: obstacle-free region $M(t)$ (white) and unknown region (grey); purple boxes are the 10 largest boxes in $M(t)$ that contain the current robot position. 42

7.5 Absence of disturbances. Computation times for Implicit CISs for different levels L of the full hierarchy, i.e., computing L Implicit CISs per level. (a) Safe sets with $2n$ constraints, $n \leq 200$. (b) Safe sets with n^2 constraints, $n \leq 100$. 45

7.6 Presence of disturbances. Safe sets with $2n$ constraints, $n \leq 20$. Computation times for Implicit RCISs. (a) Different levels L of the hierarchy. (d) Individual implicit RCIS, $\mathcal{C}_{xv,(\tau,\lambda)}$, for different values of (τ, λ). 46

7.7 Computation times for $\mathcal{C}_{xv,(0,2)}$, its projection $\mathcal{C}_{x,(0,2)}$, the methods in [TJ15] and [LTJ18], and $\mathcal{C}_{max}$. Logarithmic scale. Note: [LTJ18] is evaluated in the absence of disturbances as it considers only nominal systems. For the other methods the performance without disturbance is similar or better. 47

7.8 Two dimensional system in Brunovsky normal form. Randomly generated safe set (blue), $\mathcal{C}_{x,(1,2)} = \mathcal{C}_{max}$ (green), $\mathcal{C}_{ellips}$ from [LTJ18] (orange), and $\mathcal{C}_{lmi}$ from [TJ15] (red). 48

13.1 At each state, for a subformula ψ, x_ψ denotes $x_\psi = 1$, and $\bar{x}_\psi$ denotes $x_\psi = 0$. All states are valid initial states. The double line states are contained in the justice requirement. 99

14.1 Automated Air Traffic Control System. Left: minimum, maximum, and mean runtimes for rLTL and LTL model-checking, and time complexity blowup between rLTL and LTL. Right: Comparison between runtimes for rLTL and LTL (logarithmic scale). Computed over 24 experiments. 113

14.2 Scalability comparison: model-checking runtimes for (14.3) and (14.4) for different number of philosophers $n = 1, \ldots, 9$ (logarithmic scale). 114

A.1 All states are initial, double line states are contained in the set of justice requirements. 123

LIST OF TABLES

7.1 Absence of disturbances. Volume percentage with respect to the Maximal CIS. Algorithms: Our method computing different implicit CISs $\mathcal{C}_{xv,(\tau,\lambda)}$, the LMI method in [TJ15], and the method in [LTJ18] computing ellipsoidal CISs. (S) denotes a singleton set. 48

7.2 Presence of disturbances: volume percentage with respect to the Maximal RCIS. Algorithms: Our method computing different implicit RCISs $\mathcal{C}_{xv,(\tau,\lambda)}$ and the LMI method in [TJ15]. (S) denotes a singleton set. 49

7.3 Computation time when projecting the implicit RCIS, $\mathcal{C}_{xv,(\tau,\lambda)}$, to obtain the explicit CIS $\mathcal{C}_{x,(\tau,\lambda)} = \pi_n\left(\mathcal{C}_{xv,(\tau,\lambda)}\right)$ for various values of (τ,λ). Times are in seconds. 49

11.1 Desired negation vs. intuitionistic negation in $\mathbb{B}_5$. 73

12.1 The rLTL semantics via the ltl operator. 86

14.1 Operators of rLTL and LTL in Evrostos. 107

14.2 Occurrence frequency for each different rLTL truth value for 160 properties from the RERS benchmark. 112

ACKNOWLEDGMENTS

As my PhD chapter comes to a close, taking a look back to the past six years makes me realize how fortunate I am for the people that I shared this winding road with.

First and foremost, my advisor and mentor Paulo Tabuada. Aside from the many technical and intellectual discussions that a PhD entails, working with Paulo taught me the value of being precise, being meticulous, and substantiate any ideas even beyond the context of academia. I was fortunate to have such an attentive, enthusiastic, and more importantly patient advisor. More importantly, coming to the US my impression was that doing a PhD meant one could focus on research without external distress. The hard reality is that this is not the case for many students. Even to this day, I want to thank Paulo for taking care of any external distractions himself and for letting his students solely focus on, and stress about, doing good work. A privilege not to be underestimated! On a more personal note, Paulo did give me a chance in times of great uncertainty. I would not be here without him. Thank you Paulo!

Next, I want to thank my Doctoral Committee members. Namely, Christina Fragouli, I was extremely fortunate, not only to have you in the committee, but to work with you in many projects over the past five years, which cannot be overstated; Lieven Vandenberghe, it was a pleasure being in your class, easily one of the best I attended at UCLA, and having you in the committee; Jyotirmoy Deshmukh, it was an absolute privilege to meet you in just my second conference, and then in more of them that followed, always having stimulating conversations and taking your valuable insights to my work. Thank you all.

Of course this dissertation would not be possible without my colleagues who worked alongside me. Zexiang Liu and Necmiye Ozay, it was a pleasure collaborating with you on what led to the first part of this dissertation. Matthew Philippe and Daniel Neider, the very first people to work with other than my advisor, you helped me grow and our collaboration led to the second part of this dissertation. Thank you for all your work and patience.

As I mentioned before, a PhD program entails more than just research to worry about, especially for international students. Thankfully, the Office of Graduate Student Affairs was there for any unforeseen event. Deeona and Julio, thank you for listening to any concern and providing the best solution. The students are very fortunate to have you. And of course Ryo, you were the most kind and helpful. Every time I was assured that you had all the answers, always with the biggest smile. We were all lucky to know you. You are solely missed.

Another thank you to Dennis (Mike) Briggs. Being your TA was really a pleasure, one of the best people I have worked with. The respect and admiration of your students really speaks for itself. Keep being passionate and awesome at doing what you love!

There is not enough space here to thank my lab mates. We started off as colleagues and became more than friends – a real PhD support group! Marcus Lucas, Lucas Fraile, Matteo Marchi, Yanwen Mao, Yskandar Gas, Jonathan Bunton, we spent many late nights in the lab with conference deadlines closing in, Luigi Pannocchi, we spent even more nights fighting the robots' revolution against our algorithms, and Alimzhan Sultangazin, we even got to be roommates and survive our own bootcamp figuring out our next steps. It has been a real pleasure spending the past years with all of you in the lab, but even more doing so outside of it.

At the same time, this group would not be complete without Kaan Ege Ozgun, Anastasios Papathanasopoulos, Eden Haney, and Joshua Hannan. Special mention to Kiriaki Bouraki and Akrivi Soundia, my Greek family in CA who bring a little bit of home every time we get together. Thank you ελληνόπουλα! It is all of these people and the time and experiences shared that made the PhD time so unique.

I do not need to say much about my best friends from back home in Greece. Leandros, Kostas, Thomas, Dimitris, and Iordanis – we have known each other since before even high school. Even though we are scattered across the world, getting together is like a single day has not passed. Bros without borders. Same goes for Stavros, Katerina, and Stavros – we met in college, got nicknamed the "crew", and have been living up to it ever since.

A special thank you goes to Lauren and Lily Kiesel. Your unconditional support and faith in me pushed me all the way, making even the pandemic seem minuscule. I would not change knowing you for the world and you made LA a better place for myself one day at a time.

Finally, to my family. My parents Konstantinos Anevlavis and Konstantina Psarrou, and my sister Foteini. Know it is because of you that all this came to be. Through the limitless love, the support, the sacrifices, the twists and turns we have been through over the years. Despite the wide seas or continents separating at least one of us more times than I can count. This is for you.

VITA

2015	Diploma (Electrical and Computer Engineering) National Technical University of Athens (NTUA), Athens, Greece.
2016–2022	Graduate Student Researcher, Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, USA.
2017–2018	Teaching Assistant, Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, USA.
2019–2022	PhD Candidate, Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, USA.

PUBLICATIONS

T.Anevlavis, Z.Liu, N.Ozay and P.Tabuada, "Controlled invariant sets: implicit closed-form representations and applications," (under review).

T.Anevlavis, M.Philippe, D.Neider and P.Tabuada, "Being Correct Is Not Enough: Efficient Verification Using Robust Linear Temporal Logic," ACM Trans.Comput.Logic 23 (2).

Z.Liu, **T.Anevlavis**, N.Ozay and P.Tabuada, "Automaton-based implicit controlled invariant set computation for discrete-time linear systems," 2021 60th IEEE Conference on Decision and Control (CDC).

J.Bunton, **T.Anevlavis**, G.Verma, C.Fragouli and P.Tabuada, "Split to win: near-optimal sensor network synthesis via path-greedy subproblems," 2021 IEEE Military Communications Conference (MILCOM).

L.Pannocchi, **T.Anevlavis**, and P.Tabuada, "Trust your supervisor: quadrotor obstacle avoidance using controlled invariant sets," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

T.Anevlavis, Z.Liu, N.Ozay and P.Tabuada, "An enhanced hierarchy for (robust) controlled invariance," 2021 American Control Conference (ACC).

T.Anevlavis, J.Bunton, A.Parayil, J.George and P.Tabuada, "To beam or not to beam? Beamforming with submodularity-inspired group sparsity," 2020 59th IEEE Conference on Decision and Control (CDC).

T.Anevlavis and P.Tabuada, "A simple hierarchy for computing controlled invariant sets," 2020 23rd International Conference on Hybrid Systems: Computation and Control (HSCC).

T.Anevlavis and P.Tabuada, "Computing controlled invariant sets in two moves," 2019 IEEE 58th Conference on Decision and Control (CDC).

T.Anevlavis, D.Neider, M.Philippe, and P.Tabuada, "Evrostos: The rLTL Verifier," 2019 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC).

T.Anevlavis, M.Philippe, D.Neider and P.Tabuada, "Verifying rLTL formulas: now faster than ever before!," 2018 IEEE Conference on Decision and Control (CDC).

Part I

Robust controlled invariant sets: implicit closed-form representations and applications

CHAPTER 1

Introduction

In an increasingly autonomous world, safety has recently come under the spotlight. A safety enforcing controller is understood as one that indefinitely keeps the state of the system within a set of safe states notwithstanding the presence of uncertainties. A natural solution that guarantees safety is to initialize the state of the system inside a Robust Controlled Invariant Set (RCIS) within the set of safe states. Any RCIS is defined by the property that any trajectory starting within, can always be forced to remain therein and, hence, inside the set of safe states. Consequently, RCISs are at the core of controller synthesis for safety-critical applications.

Since the conception of the standard method for computing the Maximal RCIS of discrete-time systems [Ber72], which is known to suffer from poor scaling with the system's dimension and no guarantees of termination, numerous approaches attempted to alleviate these drawbacks. A non-exhaustive overview is found in Section 1.1.

An alternative approach is to construct an implicit representation for an RCIS. The specific implicit representation used in this paper is a set in the higher dimensional space of states and finite input sequences. We argue that in many practical, safety-critical applications, such as Model Predictive Control (MPC) and supervisory control, knowledge of the explicit RCIS is not required and the implicit representation suffices. Consequently, by exploiting the efficiency of the implicit representation the aforementioned ideas are suitable for systems with large dimensions.

In this manuscript, we propose a general framework for *computing (implicit) RCISs* for

discrete-time linear systems with additive disturbances, under polytopic state, input, and even mixed, constraints. We consider RCISs parameterized by collections of *eventually periodic* input sequences and prove that this choice leads to a closed-form expression for an implicit RCIS in the space of states and finite input sequences. Moreover, this choice results in a systematic way to obtain larger RCISs, which we term a *hierarchy*. Once the (implicit) RCIS is computed, any controller rendering the RCIS invariant can be used in practice. In the absence of disturbances, our method is complete. Even though, this property is lost in the presence of disturbances, a weak completeness result is established. Finally we study, both theoretically and experimentally, safety-critical scenarios and establish that the efficient implicit representation suffices in place of computing the exact RCIS. The content presented in this first part of this manuscript can also be found in [ALO21a].

1.1 Related literature

Several recent methods by the author and co-authors [AT19, AT20, ALO21b, WO20] develop efficient methods constructing such representations in closed-form. These approaches consider different collections of periodic input sequences and can be viewed as special instances of the method proposed here. In most of these works, the focus was on constructing the implicit representation, but always performing a projection step afterwards and working with the explicit RCIS. Thus, obtaining controlled invariant sets in two moves. Here, we not only present a more general framework that encapsulates prior work, but also demonstrate how the closed-form implicit RCISs can be used in practice.

In addition to the aforementioned methods, a plethora of works have attempted to alleviate the poor scalability and the absence of termination guarantees of the standard method for computing the Maximal CIS of discrete-time systems introduced in [Ber72]. The following list is not exhaustive.

One line of work [KHJ14, SG12, OTH19] focuses on outer and inner approximations of

the Maximal CIS by solving either LPs or QPs. The resulting sets, however, are not always invariant. Other methods compute exact ellipsoidal CISs efficiently and, thus, offer improved scalability, such as [LTJ18] which solves Semi-Definite Programs (SDP) for a class of hybrid systems. Nevertheless, the resulting ellipsoidal sets are generally small. In addition, for online control problems, like MPC and supervisory control, polytopic sets are preferred to ellipsoids. The reason is that polytopes result in either LPs or QPs, which are solved more efficiently compared to Quadratically Constrained Quadratic Programs (QCQP) that stem from ellipsoids.

In the presence of bounded disturbances, when the set of safe states are polytopes, [RT17] computes inner and outer approximations of the Maximal RCIS for linear systems.

Ideas similar to ours, in the sense that finite input sequences are used, have been explored in the context of MPC [MSR05]. The goal there is to establish asymptotic stability of a linear system, whereas in our case we exploit finite input sequences that describe the proposed control behavior, which leads to a closed-form expression for an implicit representation of controlled invariant sets.

Many popular approaches first close the loop with a linear state-feedback control law, and then compute the Robust Positively Invariant Set (RPIS) of the closed-loop system. Under this umbrella, an idea close to ours is found in [LS15], where recurrent sets are computed in the context of MPC for the closed-loop system in the absence of disturbances. This can be understood as a special case of our eventually periodic approach. In the case of additive input disturbance, [MOB18] computes RPISs for the closed-loop system and enlarge the controllable region by use of an interpolation method.

In a similar spirit, i.e., by restricting to linear state-feedback control laws, the following works focus on reducing the computational cost and employ iterative procedures to compute low-complexity or fixed-complexity RCISs and their associated feedback gains. In [TJ15] low-complexity RCISs are found via Semi-Definite Programming (SDP) for systems with norm-bounded uncertainties. More recently, [GKF19, GF19] compute low- and fixed-complexity

RCISs respectively for systems with rational parameter dependence. The complexity in [GKF19] is twice the number of states, while [GF19] is more flexible as the complexity can be pre-decided.

The aforementioned group of methods is typically conservative as only linear state-feedback controllers are considered.

The work of [MHO18] computes larger controlled contractive sets of specified degree for nominal linear systems by solving Sum Of Squares (SOS) problems, but to do so prior knowledge of a contractive set is required. The scalability of this approach is limited by the size of the SOS problems that can be handled, and so is its extension to systems with polytopic uncertainty as it significantly increases the size of the SOS problem [MHO18].

CHAPTER 2

Problem formulation

Notation: Let $\mathbb{R}$ and $\mathbb{N}$ denote the set of real numbers and positive integers respectively. Given two sets $P, Q \subset \mathbb{R}^n$, we denote by $P + Q = \{x \in \mathbb{R}^n \mid x = p + q, p \in P, q \in Q\}$ their Minkowski sum and by $Q - P = \{x \in \mathbb{R}^n \mid x + p \in Q, p \in P\}$ their Minkowsky difference. By slightly abusing the notation, we denote the Minkowsky sum of a singleton $\{x\}$ and a set P by $x + P$. We denote a block-diagonal matrix M with blocks $M_1, \ldots, M_N$ by $M = \text{blkdiag}(M_1, \ldots, M_N)$. A set $S \subset \mathbb{R}^n$ is convex if $\forall x, y \in S, (1-t)x + ty \in S, \forall t \in [0, 1]$. Then, given a set $P \subset \mathbb{R}^n$, we define the convex hull of P, $\text{conv}(P)$, as the smallest convex set enclosing P. Formally, $\text{conv}(P) = \{\sum_{i=1}^k \lambda_i p_i | \lambda_i \geq 0, p_i \in P, i = 1, \ldots, k, \text{ and } \sum_{i=1}^k \lambda_i\}$. Moreover, given a matrix $A \in \mathbb{R}^{m \times n}$ and a set $P \in \mathbb{R}^n$, we denote the linear transformation of P through A by $AP = \{Ax \in \mathbb{R}^m \mid x \in P\}$. Given a set $S \subset \mathbb{R}^n \times \mathbb{R}^m$, we denote its projection onto the first n coordinates by $\pi_n(S)$. Finally, let $\mathbb{I}$ and $\mathbb{0}$ denote the identity and zero matrices of appropriate sizes respectively, while similarly $\mathbb{1}$ denotes a vector with all entries equal to 1.

Let us begin by providing the necessary definitions.

Definition 1 (Discrete-time linear system)**.** *A* Discrete-Time Linear System *(DTLS)* Σ *is a linear difference equation:*

$$x^+ = Ax + Bu + Ew, \tag{2.1}$$

where $x \in \mathbb{R}^n$ *is the state of the system,* $u \in \mathbb{R}^m$ *is the input, and* $w \in W \subseteq \mathbb{R}^d$ *is a disturbance term. Moreover, we have that* $A \in \mathbb{R}^{n \times n}$*,* $B \in \mathbb{R}^{n \times m}$*, and* $E \in \mathbb{R}^{n \times d}$*.*

Definition 2 (Polytope)**.** *A* polytope $S \subset \mathbb{R}^n$ *is a bounded set of the form:*

$$S = \{x \in \mathbb{R}^n \mid Gx \leq f\}, \tag{2.2}$$

where $G \in \mathbb{R}^{k \times n}$*,* $f \in \mathbb{R}^k$ *for some* $k > 0$*.*

Assumption 1. *In this first part of the manuscript we focus on systems and safe sets that satisfy the following:*

1) There exists a suitable state feedback transformation that makes the matrix A *of system* Σ nilpotent*. For a nilpotent matrix, there exists a* $\nu \in \mathbb{N}$ *such that* $A^\nu = 0$*.*

2) The safe set $S_{xu} \subset \mathbb{R}^n \times \mathbb{R}^m$*, i.e., the set defining the state-input constraints for* Σ*, is a polytope.*

3) The disturbance set $W \subset \mathbb{R}^d$ *is a polytope.*

Remark 1. *Notice that Assumption 1 is satisfied by any controllable system, as for any such system there exists a feedback gain* $K \in \mathbb{R}^{m \times n}$ *such that* $A + BK$ *is nilpotent. Thus, any controllable system satisfies Assumption 1 by pre-feedbacking the system with* $u = Kx + v$ *and taking* v *as the new control input [AM06, Ch.3]. In this case, the nilpotency index* ν *is equal to the largest controllability index of* Σ*.*

Given Remark 1, for the remainder of this first part of the manuscript we assume that A is already nilpotent.

Remark 2. *Our goal is to compute a representation of a controlled invariant set. To that end, the pre-feedback transformation discussed above provides a convenient mathematical tool for obtaining this representation. This transformation simply moves the original input constraints to the transformed state-input space and, thus, it neither changes the original problem nor restricts the control authority. Once we compute the controlled invariant set, we can always apply the inverse transformation and obtain its representation in the original state-input space.*

Definition 3 (Robust Controlled Invariant Set)**.** *Given a DTLS* Σ *and an associated safe set* $S_{xu} \subset \mathbb{R}^n \times \mathbb{R}^m$*, a set* $\mathcal{C} \subset \mathbb{R}^n$ *is a* Robust Controlled Invariant Subset *for* Σ *within* $\pi_n(S_{xu})$ *if:*

$$x \in \mathcal{C} \Rightarrow \exists u \in \mathbb{R}^m, \text{with } (x,u) \in S_{xu}, \text{ such that}$$
$$\forall w \in W,\ Ax + Bu + Ew \in \mathcal{C}.$$

The main goal of this first part of the manuscript is to compute an *implicit representation of an RCIS in closed-form.* Hereafter, we refer to this representation as the *implicit RCIS.*

Definition 4 (Implicit RCIS)**.** *Given a DTLS* Σ*, a safe set* $S_{xu} \subset \mathbb{R}^n \times \mathbb{R}^m$*, and an input sequence* $v : \{1, \ldots, q\} \to \mathbb{R}^m$ *of length* q*, i.e.,* $v \in \mathbb{R}^{qm}$*, a set* $\mathcal{C}_{xv} \subset \mathbb{R}^n \times \mathbb{R}^{qm}$ *is an* Implicit RCIS *for* Σ *if* $\pi_n(\mathcal{C}_{xv}) \subseteq \pi_n(S_{xu})$ *is an RCIS for* Σ*.*

The following result stems directly from Definition 3.

Proposition 2.0.1. *The union and the convex hull of two robust controlled invariant sets are robustly controlled invariant.*

For dynamical systems, i.e., systems Σ as in (2.1) where $B = 0$, the analogous concept to RCISs is defined below.

Definition 5 (Robust Positively Invariant Set)**.** *Given a dynamical system* Σ *defined as* $\Sigma : x^+ = Ax + Ew$ *and a safe set* $S_x \subset \mathbb{R}^n$*, a set* $\mathcal{C} \subset \mathbb{R}^n$ *is a* Robust Positively Invariant Subset *for* Σ *within* S *if:*

$$x \in \mathcal{C} \Rightarrow \forall w \in W,\ Ax + Ew \in \mathcal{C}.$$

We define the *accumulated disturbance set* at time t by:

$$\overline{W}_t = \sum_{i=1}^{t} A^{i-1} EW. \tag{2.3}$$

By nilpotency of A we have that:

$$\overline{W}_\infty = \sum_{i=1}^{\infty} A^{i-1}EW = \sum_{i=1}^{\nu} A^{i-1}EW. \tag{2.4}$$

In the literature, $\overline{W}_\infty$ is called the *Minimal RPIS* of the system $x^+ = Ax + Ew$ [RKK05].

Next, we introduce an operator used throughout this first part of the manuscript.

Definition 6 (Reachable set)**.** *Given a DTLS Σ and a set $X \subset \mathbb{R}^n$, define the* reachable set *from X under input sequence $\{u_i\}_{i=0}^{t-1}$ as:*

$$\mathcal{R}_\Sigma\big(X, \{u_i\}_{i=0}^{t-1}\big) = A^t X + \sum_{i=1}^{t} A^{i-1}Bu_{t-i} + \overline{W}_t. \tag{2.5}$$

Intuitively, $\mathcal{R}_\Sigma\big(X, \{u_i\}_{i=0}^{t-1}\big)$ maps a set X and an input sequence $\{u_i\}_{i=0}^{t-1}$ to the set of all states that can be reached from X in t steps when applying said input sequence. Conventionally, $\mathcal{R}_\Sigma(X, \emptyset) = X$ and when X is a singleton, i.e., $X = \{x\}$, we abuse notation to write $\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big)$.

CHAPTER 3

Implicit representation of controlled invariant sets for linear systems

The classical algorithm that computes the *Maximal* RCIS consists of an iterative procedure [Ber72, DC71] and theoretically works for any discrete-time system and safe set. However, this approach is known to suffer from the curse of dimensionality and its termination is not guaranteed.

To alleviate these drawbacks, we propose an algorithm that is guaranteed to terminate and computes an implicit RCIS efficiently in closed-form, thus being suitable for high dimensional systems. Moreover, by optionally projecting the implicit RCIS back to the original state-space one computes an explicit RCIS. Overall, the proposed algorithm computes controlled invariant sets in one and two moves respectively.

The goal of this section is to present a *finite implicit representation* of an RCIS. That is, we provide a *closed-form* expression for an *implicit RCIS* characterized by constraints on the state and on a finite input sequence, whose length is the design parameter. This results in a polytopic RCIS in a higher dimensional space. Intuitively, the implicit RCIS contains the pairs of states and appropriate finite input sequences that guarantee that the state remains in the safe set indefinitely.

3.1 General implicit robust controlled invariant sets

We begin by discussing the general construction of a polytopic implicit RCIS. Our goal is to parameterize RCISs by collections of input sequences. Thus, we consider inputs u_t to Σ that evolve as the output of a linear dynamical system, Σ_C, whose state is a *sequence of q inputs, v*, i.e.:

$$\Sigma_C \quad : \quad \begin{aligned} v_{t+1} &= Pv_t, \\ u_t &= Hv_t, \end{aligned} \tag{3.1}$$

where $v \in \mathbb{R}^{mq}$, $P \in \mathbb{R}^{mq \times mq}$, and $H \in \mathbb{R}^{m \times mq}$. The choice of a linear dynamical system stems from our safe set being a polytope per Assumption 1. By using system Σ_C we preserve the linearity of the safe set constraints and we are, hence, able to compute polytopic RCISs within polytopic safe sets. The resulting input to Σ can be expressed as:

$$u_t = Hv_t = HP^t v_0, \tag{3.2}$$

for an initial choice of $v_0 \in \mathbb{R}^{mq}$. We can then lift system Σ, after closing the loop with Σ_C, to the following companion dynamical system:

$$\Sigma_{xv} \quad : \quad \begin{bmatrix} x^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} A & BH \\ \mathbb{0} & P \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} E \\ \mathbb{0} \end{bmatrix} w. \tag{3.3}$$

Given the safe set S_{xu}, we construct the companion safe set S_{xv} defined as follows, $S_{xv} = \{(x, v) \in \mathbb{R}^n \times \mathbb{R}^{mq} \mid (x, Hv) \in S_{xu}\}$. The companion system of (2.1) is the closed-loop dynamics of (2.1) with a control input in (3.2). Then, the companion safe set simply constrains the closed-loop state-input pairs in the original safe set, i.e., $(x_t, Hv_t) \in S_{xu}$.

Theorem 3.1.1 (Generalized implicit RCIS)**.** *Let $\mathcal{C}_{xv}$ be an RPIS of the companion system Σ_{xv} within the companion safe set S_{xv}. The projection of $\mathcal{C}_{xv}$ onto the first n coordinates, $\pi_n(\mathcal{C}_{xv})$, is an RCIS of the original system Σ within S_{xu}. In other words, $\mathcal{C}_{xv}$ is* an implicit RCIS *of Σ.*

Proof. Let $x \in \pi_n(\mathcal{C}_{xv})$. Then, there exists a $v \in \mathbb{R}^{mq}$ such that $(x, v) \in \mathcal{C}_{xv}$. Define $u = Hv$ and pick an arbitrary $w \in W$. By construction of S_{xv}, $(x, u) \in S_{xu}$. Since $\mathcal{C}_{xv}$ is an RPIS, we have that $(x^+, v^+) = (Ax + Bu + Ew, Pv) \in \mathcal{C}_{xv}$ and thus $x^+ \in \pi_n(\mathcal{C}_{xv})$. By Definition 3, $\pi_n(\mathcal{C}_{xv})$ is an RCIS of Σ in S_{xu}. □

In what follows, we study the conditions on P and H such that the Maximal RPIS of Σ_{xv} is represented in closed-form.

3.2 Finite reachability constraints

By definition of the companion safe set S_{xv} and Definition 5, we have that any state (x, v) belongs to the Maximal RPIS of Σ_{xv} within S_{xv}, if and only if, the input sequence $\{u_i\}_{i=0}^{t-1}$, with each input as in (3.2), satisfies:

$$\left(\mathcal{R}_\Sigma\left(x, \{u_i\}_{i=0}^{t-1}\right), u_t\right) \subseteq S_{xu},\ t \geq 0, \tag{3.4}$$

where $\mathcal{R}_\Sigma\left(x, \{u_i\}_{i=0}^{t-1}\right) \subseteq \mathbb{R}^n$, $u_t \in \mathbb{R}^m$, and the pair $\left(\mathcal{R}_\Sigma\left(x, \{u_i\}_{i=0}^{t-1}\right), u_t\right) \subseteq \mathbb{R}^n \times \mathbb{R}^m$. By Theorem 3.1.1, the above constraints characterize the states and input sequences within an implicit RCIS of Σ, such that the pair (x, u) stays inside the safe set S_{xu} indefinitely. Notice that (3.4) defines an *infinite* number of constraints in general. In this section, we investigate under what conditions we can reduce the above constraints into a *finite* number and compute them explicitly. Then, we use these constraints to construct the promised implicit RCIS.

Definition 7 (Eventually periodic behavior)**.** *Consider two integers $\tau \in \mathbb{N} \cup \{0\}$ and $\lambda \in \mathbb{N}$. A control input u_t follows an* eventually periodic behavior *if:*

$$u_{t+\lambda} = u_t,\ \textit{for all } t \geq \tau. \tag{3.5}$$

We call τ the transient *and λ the* period*.*

Proposition 3.2.1 (Finite reachability constraints)**.** *Consider a DTLS Σ satisfying Assumption 1. If the input u_t follows an eventually periodic behavior with transient $\tau \in \mathbb{N} \cup \{0\}$*

and period $\lambda \in \mathbb{N}$, then the infinite constraints in (3.4) *are reduced to a finite number of constraints.*

Proof. Under Assumption 1 the matrix A is nilpotent with nilpotency index ν. Consequently, given (2.5), the reachable set from a state x for $t \geq \nu$ depends only on the past ν inputs:

$$\mathcal{R}_\Sigma\big(\{u_i\}_{i=0}^{t-1}\big) = \sum_{i=1}^{\nu} A^{i-1}Bu_{t-i} + \overline{W}_\infty,\ t \geq \nu, \tag{3.6}$$

where we abuse notation and write $\mathcal{R}_\Sigma\big(\{u_i\}_{i=0}^{t-1}\big)$, omitting the state x, to denote dependency only on the inputs. Then, for $t \geq \nu + \tau$:

$$\mathcal{R}_\Sigma\big(\{u_i\}_{i=0}^{t-1}\big) = \sum_{i=1}^{\nu} A^{i-1}Bu_{t-i} + \overline{W}_\infty \overset{(3.5)}{=} \sum_{i=1}^{\nu} A^{i-1}Bu_{t+\lambda-i} + \overline{W}_\infty = \mathcal{R}_\Sigma\big(\{u_i\}_{i=0}^{t+\lambda-1}\big)\,.$$

Therefore, under inputs with eventually periodic behavior the reachability constraints repeat themselves after $t = \nu + \tau + \lambda$. As a result, we can split the constraints in (3.4) as:

$$\big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu},\ t = 0, \ldots, \nu - 1, \tag{3.7}$$

$$\big(\mathcal{R}_\Sigma\big(\{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu},\ t = \nu, \ldots, \nu + \tau + \lambda - 1. \tag{3.8}$$

The above suggests that $\big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu}$ for all $t \geq 0$ can be replaced with only $\nu + \tau + \lambda$ constraints. □

Proposition 3.2.1 provides a finite representation of the constraints in (3.4) under the eventually periodic input behavior in (3.5). The next question we address concerns characterizing the classes of policies that guarantee the behavior in (3.5).

3.3 Implicit robust controlled invariant sets in closed-form

Recall that our goal is to derive a closed-form expression for an implicit RCIS of Σ, which is essentially the Maximal RPIS of the companion system Σ_{xv} by Theorem 3.1.1. So far we proved that, in general, inputs with eventually periodic behavior result in finite reachability

constraints. Clearly, the parameterized input in (3.2) follows an eventually periodic behavior as in (3.5) if:

$$P^t = P^{t+\lambda},\ t \geq \tau, \tag{3.9}$$

i.e., P is an *eventually periodic* matrix with transient τ and period λ.

Proposition 3.3.1 (Structure of eventually periodic matrices)**.** *Any* eventually periodic *matrix $P \in \mathbb{R}^{n\times n}$ has eigenvalues that are either* 0 *or λ-th roots of unity. If $\tau \neq 0$, i.e., P is not purely periodic, then P has at least one* 0 *eigenvalue with algebraic multiplicity equal to τ and geometric multiplicity equal to* 1*. If $P^\tau \neq 0$, i.e., P is not nilpotent, then P has at least one eigenvalue that is a λ-th root of unity.*

Proof. Let $\mathrm{v} \neq 0$ be an eigenvector of P and δ the corresponding eigenvalue, i.e., $P\mathrm{v} = \delta\mathrm{v}$. Then, (3.9) for $t \geq \tau$ yields:

$$P^t = P^{t+\lambda} \Rightarrow P^t\mathrm{v} = P^{t+\lambda}\mathrm{v} \Leftrightarrow \delta^t\mathrm{v} = \delta^{t+\lambda}\mathrm{v} \overset{\mathrm{v}\neq 0}{\Leftrightarrow} \delta^t = \delta^{t+\lambda} \Leftrightarrow \delta^t\left(1-\delta^\lambda\right) = 0,$$

that is, the eigenvalues δ of P are only 0 or λ-th roots of unity.

Consider now the Jordan normal form $P = MJM^{-1}$ [Lau04]. This form is unique up to the order of the Jordan blocks, and $P^t = MJ^tM^{-1}$. Without loss of generality, we write:

$$J = \begin{bmatrix} J_1 & \mathbb{0} \\ \mathbb{0} & J_2 \end{bmatrix},$$

where J_1 is the Jordan block corresponding to the eigenvalues of P that are 0, and J_2 is the Jordan block corresponding to the eigenvalues of P that are the λ-th roots of unity. Thus, J_1 is nilpotent. Then, when $\tau \neq 0$, equality (3.9) is equivalent to:

$$P^t = P^{t+\lambda} \Leftrightarrow MJ^tM^{-1} = MJ^{t+\lambda}M^{-1},\ t \geq \tau.$$

Matrix J_1 vanishes in exactly τ steps, i.e., $J_1^\tau = 0$ and $J_1^t \neq 0$, for $t < \tau$. This implies that P has at least one 0 eigenvalue with algebraic multiplicity equal to τ and geometric multiplicity

equal to 1, but no 0 eigenvalues of geometric multiplicity 1 and algebraic multiplicity greater than τ.

Moreover, when P is not nilpotent, i.e., $P^\tau \neq \mathbb{0}$, for $t \geq \tau$:

$$J^t = J^{t+\lambda} \overset{J_1^t=\mathbb{0}, t\geq\tau}{\Leftrightarrow} \begin{bmatrix} \mathbb{0} & \mathbb{0} \\ \mathbb{0} & J_2^t \end{bmatrix} = \begin{bmatrix} \mathbb{0} & \mathbb{0} \\ \mathbb{0} & J_2^{t+\lambda} \end{bmatrix} \Leftrightarrow J_2^t = J_2^{t+\lambda}.$$

Thus, P has at least one eigenvalue that is a λ-th root of unity. □

Corollary 3.3.2. *The class of matrices described by Proposition 3.3.1 that satisfies* (3.9) *can be written, up to a similarity transformation, in the following form:*

$$P = \begin{bmatrix} N & Q \\ \mathbb{0} & R \end{bmatrix}, \tag{3.10}$$

where N is a nilpotent matrix with nilpotency index τ, R is a matrix whose eigenvalues are all λ-th roots of unity, i.e., $R^\lambda = \mathbb{I}$, and Q is an arbitrary matrix.

Proposition 3.3.1 and Corollary 3.3.2 guide the designer to effortlessly select matrix P via its eigenvalues or its submatrices. Moreover, it is reasonable to select the projection matrix H to be *surjective* in order to obtain a non-trivial input in (3.2).

We now show that we can compute the desired *closed-form* expression for an implicit RCIS parameterized by collections of *eventually periodic* input sequences.

Theorem 3.3.3 (Closed-form implicit RCIS)**.** *Consider a DTLS Σ and a safe set S_{xu} for which Assumption 1 holds. Select an eventually periodic matrix $P \in \mathbb{R}^{mq\times mq}$ and a surjective projection matrix $H \in \mathbb{R}^{m\times mq}$. An* implicit RCIS *for Σ within S_{xu}, denoted by $\mathcal{C}_{xv}$, is defined by the constraints:*

$$\begin{aligned} \left(A^t x + \sum_{i=1}^{t} A^{i-1} BHP^{t-i} v, HP^t v\right) &\subseteq S_{xu} - \overline{W}_t \times \{0\},\ t = 0, \ldots, \nu-1, \\ \left(\sum_{i=1}^{\nu} A^{i-1} BHP^{t-i} v, HP^t v\right) &\subseteq S_{xu} - \overline{W}_\infty \times \{0\},\ t = \nu, \ldots, \nu+\tau+\lambda-1. \end{aligned} \tag{3.11}$$

That is, the set $\mathcal{C}_{xv} \subset \mathbb{R}^n \times \mathbb{R}^{mq}$*:*

$$\mathcal{C}_{xv} = \{(x, v) \in \mathbb{R}^n \times \mathbb{R}^{mq} \mid (x, v) \text{ satisfy } (3.11)\}, \tag{3.12}$$

is computed in closed-form. Moreover, $\mathcal{C}_{xv}$ *is the Maximal RPIS of the companion dynamical system in* (3.3)*.*

Proof. By Proposition 3.2.1, the set $\mathcal{C}_{xv}$ defined by (3.11) in closed-form satisfies the constraints in (3.4) and, thus, is the Maximal RPIS of the companion system Σ_{xv} in S_{xv}. Then, by Theorem 3.1.1, C_{xv} is an implicit RCIS of Σ in S_{xu}. □

Theorem 3.3.3 provides an implicit RCIS, $\mathcal{C}_{xv}$, in closed-form. This set defines pairs of states and finite input sequences such that the state remains in the safe set indefinitely.

Remark 3 (On the choice of input behavior)**.** *Notice that the open-loop eventually periodic policy used to parameterize the implicit RCIS is only a means towards its computation in closed-form. In practice, after computing an RCIS, we can use any controller appropriate for the task at hand. This is illustrated in our case studies in Section 7, where the controller of the system is independent of the RCIS implicit representation. For instance, once an RCIS is available one defines a closed-loop non-periodic and memoryless controller* $K : \mathbb{R}^n \to \mathbb{R}^m$ *for which* $Ax + BK(x)$ *belongs to the RCIS when* x *is an element of the RCIS.*

3.4 Obtaining an explicit robust controlled invariant set

Finally, one computes an explicit RCIS as the projection of $\mathcal{C}_{xv}$ onto the first n coordinates.

Proposition 3.4.1 (Computation of explicit RCIS)**.** *Consider a DTLS* Σ *and a safe set* S_{xu} *for which Assumption 1 holds. Select an eventually periodic matrix* $P \in \mathbb{R}^{mq \times mq}$ *and a projection matrix* $H \in \mathbb{R}^{m \times mq}$ *for* Σ_C*. An* explicit RCIS *for* Σ *within* $\pi_n(S_{xu})$*, denoted by* $\mathcal{C}_x$*, is computed as:*

$$\mathcal{C}_x = \pi_n(\mathcal{C}_{xv}), \tag{3.13}$$

where $\mathcal{C}_{xv}$ is the implicit RCIS given in closed-form in (3.12).

Proof. By Theorem 3.1.1, $C_x = \pi_n(\mathcal{C}_{xv})$ is an RCIS. □

Notice that (3.13) requires a projection, which is an expensive operation. The size of the lifted space leads to a trade-off: on the one hand it can result to larger RCISs, but on the other it requires more effort if the optional projection step is taken. As numerical examples show in subsequent sections, this single projection step is in general more efficient than the classical algorithm that performs projections during each iteration.

CHAPTER 4

A hierarchy of controlled invariant sets

Our main result, Theorem 3.3.3, provides a closed-form expression for an implicit RCIS, $\mathcal{C}_{xv}$, with constraints on the state of the system and on a finite sequence of inputs. The resulting sets depend on the choice of the projection matrix H and the eventually periodic matrix P in (3.1).

In this section, we show that we can *systematically* construct a sequence of RCISs that form a *hierarchy*, i.e., a non-decreasing sequence. Our goal is to provide a closed-form expression for the implicit RCISs corresponding to this hierarchy. Towards this, we define a special case of eventually periodic input behaviors, which we term as *lasso input behaviors.*

Definition 8 ((τ, λ)-lasso input behavior)**.** *Consider two integers $\tau \in \mathbb{N} \cup \{0\}$ and $\lambda \in \mathbb{N}$, and let $L = \tau + \lambda$. A control input generated by the dynamical system Σ_C in* (3.1) *follows a* (τ, λ)-lasso behavior *if the corresponding P and H matrices in Σ_C are block diagonal:*

$$\begin{aligned} P_{(\tau,\lambda)} &= \text{blkdiag}\left(\bar{P}, \ldots, \bar{P}\right) \in \mathbb{R}^{mL \times mL}, \\ H_{(\tau,\lambda)} &= \text{blkdiag}\left(\bar{H}, \ldots, \bar{H}\right) \in \mathbb{R}^{m \times mL}, \end{aligned} \tag{4.1}$$

with m blocks each and $\bar{P}$, $\bar{H}$ defined as:

$$\begin{aligned} \bar{P} &= \begin{bmatrix} \mathbb{0} & & \mathbb{I} & & \\ 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{L \times L}, \\ \bar{H} &= \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{R}^{1 \times L}. \end{aligned} \tag{4.2}$$

In the last row of $\bar{P}$ the 1 occurs at the τ-th position. It is easy to verify that $P_{(\tau,\lambda)}$ in (4.1) *is of the form* (3.10)*. A (τ, λ)-lasso behavior has a transient of τ inputs followed by periodic inputs with period λ.*

As a consequence of Theorems 3.3.3 and 3.4.1, when selecting the matrices $P_{(\tau,\lambda)}$ and $H_{(\tau,\lambda)}$ as in (4.1), we obtain an implicit RCIS, $\mathcal{C}_{xv,(\tau,\lambda)}$ in closed-form, and an explicit RCIS, $\mathcal{C}_{x,(\tau,\lambda)}$, as its projection. We utilize the (τ,λ)-lasso behavior to formalize a hierarchy of RCISs with a single decision parameter L.

Definition 9 (Lassos of same length)**.** *Select $L \in \mathbb{N}$. We define the set of all pairs $(\tau,\lambda) \in \mathbb{N} \cup \{0\} \times \mathbb{N}$ corresponding to lassos of length L as:*

$$\Theta_L = \{(\tau,\lambda) \in \mathbb{N} \cup \{0\} \times \mathbb{N} \mid \tau + \lambda = L\}. \tag{4.3}$$

The cardinality of Θ_L is exactly L.

The next result provides a way to systematically construct implicit RCISs in closed-form such that the corresponding explicit RCISs form a hierarchy.

Theorem 4.0.1 (Hierarchy of RCISs)**.** *Consider a DTLS Σ and a safe set S_{xu} for which Assumption 1 holds, and select an integer $L \in \mathbb{N}$. Given L, the set $\mathcal{C}_{xv,\mathrm{L}} \subset \mathbb{R}^n \times \mathbb{R}^{mL}$:*

$$\mathcal{C}_{xv,\mathrm{L}} = \bigcup_{(\tau,\lambda)\in\Theta_L} \mathcal{C}_{xv,(\tau,\lambda)}, \tag{4.4}$$

is the implicit RCIS induced by the L-level of the hierarchy, where each $\mathcal{C}_{xv,(\tau,\lambda)}$ is computed in closed-form in (3.12) *with P and H as in* (4.1)*. In addition, the explicit RCIS:*

$$\mathcal{C}_{x,\mathrm{L}} = \pi_n(\mathcal{C}_{xv,\mathrm{L}}) = \bigcup_{(\tau,\lambda)\in\Theta_L} \pi_n\left(\mathcal{C}_{xv,(\tau,\lambda)}\right) = \bigcup_{(\tau,\lambda)\in\Theta_L} \mathcal{C}_{x,(\tau,\lambda)}, \tag{4.5}$$

corresponding to the L-level of the hierarchy contains any RCIS lower in the hierarchy, i.e.:

$$\mathcal{C}_{x,\mathrm{L}} \supseteq \mathcal{C}_{x,\mathrm{L}'}, \text{ for any } L, L' \in \mathbb{N} \text{ with } L' < L. \tag{4.6}$$

Proof. First, the sets $\mathcal{C}_{xv,\mathrm{L}}$ and $\mathcal{C}_{x,\mathrm{L}}$ are implicit and explicit RCISs respectively as the unions of, implicit and explicit, RCISs by Proposition 2.0.1. Next we prove (4.6) for the case of L and $L+1$, while the more general statement follows by a simple induction argument.

For any $\lambda \in \mathbb{N}$ such that $(\tau, \lambda) \in \Theta_L$, we have by (4.3) that $(\tau+1, \lambda) \in \Theta_{L+1}$. It is easy to conceptualize that:

$$\mathcal{C}_{x,(\tau+1,\lambda)} \supseteq \mathcal{C}_{x,(\tau,\lambda)}, \tag{4.7}$$

as a (τ, λ)-lasso input behavior can always be embedded in a $(\tau+1, \lambda)$-lasso input. Notice now that from (4.5) it follows that:

$$\begin{aligned}\mathcal{C}_{x,(\mathrm{L}+1)} &= \bigcup_{(\tau,\lambda)\in\Theta_{L+1}} \mathcal{C}_{x,(\tau,\lambda)} = \left(\bigcup_{(\tau,\lambda)\in\Theta_L} \mathcal{C}_{x,(\tau+1,\lambda)}\right) \bigcup \mathcal{C}_{x,(0,L+1)} \\ &\overset{(4.7)}{\supseteq} \left(\bigcup_{(\tau,\lambda)\in\Theta_L} \mathcal{C}_{x,(\tau,\lambda)}\right) \bigcup \mathcal{C}_{x,(0,L+1)} \overset{(4.5)}{=} \mathcal{C}_{x,\mathrm{L}} \bigcup \mathcal{C}_{x,(0,L+1)}.\end{aligned}$$

The above entails that $\mathcal{C}_{x,(\mathrm{L}+1)} \supseteq \mathcal{C}_{x,\mathrm{L}}$. □

Corollary 4.0.2. *Using the standard big-M formulation [Lof12] we can write the implicit RCIS $\mathcal{C}_{xv,\mathrm{L}}$ in closed-form as:*

$$\mathcal{C}_{xv\zeta,\mathrm{L}} = \left\{ (x, v, \zeta) \middle| \sum_{i=1}^{L} \zeta_i = 1, G_i(x, v) \leq f_i + (1-\zeta_i) M \mathbb{1} \right\}, \tag{4.8}$$

where $\zeta \in \{0,1\}^L$, G_i and f_i describe each of the L polytopes $\mathcal{C}_{xv,(\tau,\lambda)}$ in (4.4), and $M \in \mathbb{R}_+$ is sufficiently large. The set $\mathcal{C}_{xv\zeta,\mathrm{L}}$ is a polytope in $\mathbb{R}^n \times \mathbb{R}^{mL} \times \{0,1\}^L$, and its projection on $\mathbb{R}^n \times \mathbb{R}^{mL}$ is exactly the union in (4.4), while its projection on $\mathbb{R}^n$ is exactly the explicit RCIS in (4.5).

Theorem 4.0.1 defines the promised hierarchy and provides us with an implicit RCIS for each level of the hierarchy that can also be computed in closed-form in (4.8) at the cost of a lift to a higher dimensional space. Fig. 4.1 illustrates the relation in (4.6), that is, how the sets induced by each hierarchy level contain the ones induced by lower hierarchy levels.

Remark 4 (Convex hierarchy)**.** *We can replace the union in (4.4) by the convex hull* $\operatorname{conv}\left(\bigcup_{(\tau,\lambda)\in\Theta_L} \mathcal{C}_{xv,(\tau,\lambda)}\right)$. *Then, in an analogous manner, all the above results go through resulting in a* hierarchy of convex RCISs. *Similarly to (4.8), by standard set-lifting techniques, one obtains a closed-form expression for the convex hull.*

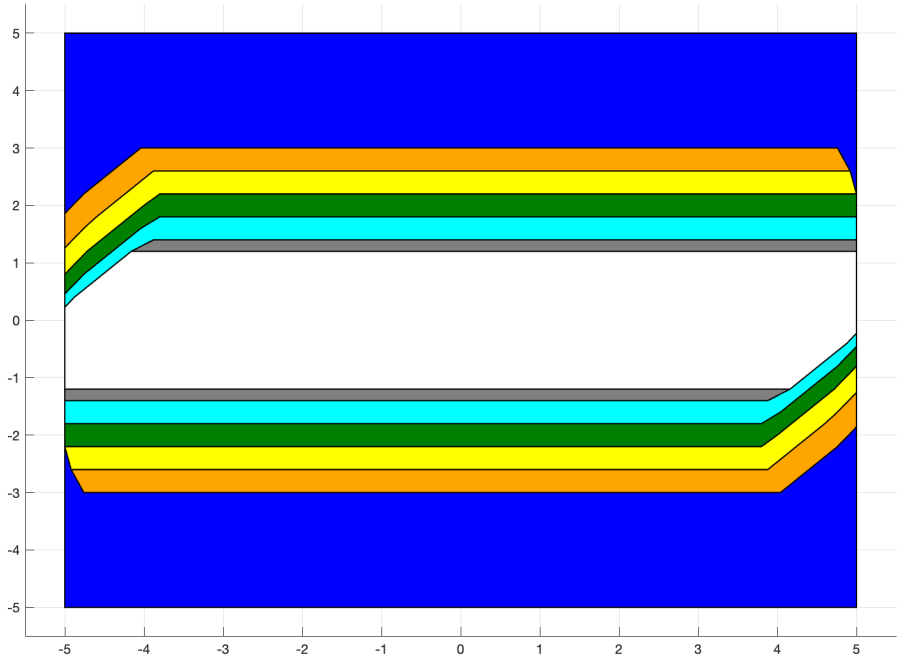


Figure 4.1: RCIS corresponding to $L = 1$ (white), $L = 2$ (gray), $L = 3$ (teal), $L = 4$ (green), $L = 5$ (yellow), and $L = 6$ (orange) for a double integrator. Safe set in blue.

Remark 5 (Partial hierarchies without union)**.** *The proposed hierarchy involves handling a union of sets either in* (4.4) *or in closed-form in* (4.8) *by introducing an additional lift. However, one might prefer to avoid unions of sets and rather work with a single convex set. In this case, notice that each set $\mathcal{C}_{xv,(\tau,\lambda)}$ involved in the hierarchy is also an implicit RCIS computed in closed-form by Theorem 3.3.3. Two more refined guidelines for obtaining larger sets, based on the choice of (τ, λ), are the following:*

1. *Given any $\lambda \in \mathbb{N}$, it holds that $\mathcal{C}_{x,(\tau+1,\lambda)} \supseteq \mathcal{C}_{x,(\tau,\lambda)}$ for any $\tau \in \mathbb{N} \cup \{0\}$.*

2. *Given any $\tau \in \mathbb{N} \cup \{0\}$, it holds that $\mathcal{C}_{x,(\tau,\lambda)} \supseteq \mathcal{C}_{x,(\tau,\lambda')}$ for any $\lambda, \lambda' \in \mathbb{N}$ such that $\lambda = k\lambda'$, $k \in \mathbb{N}$, i.e., λ is a multiple of λ', see [AT20, Section 4.6] when $\tau = 0$.*

The above can direct the designer in search of larger RCISs that are based on single implicit RCIS.

CHAPTER 5

Performance bound for the proposed method

Numerical examples, to be presented later, will show that the projection of the proposed implicit RCIS onto the original state-space can coincide with the Maximal RCIS. However, this is not always the case. When there is a gap between our projected set and the Maximal RCIS, one may wonder if that gap is fundamental to our method. In other words, can we arbitrarily approximate the Maximal RCIS with the projection of our implicit RCIS by choosing better P and H matrices?

In this section we aim to answer the above question and provide insights into the completeness of our method. Given (2.4), we define the nominal DTLS $\overline{\Sigma}$ and the associated nominal safe set $\overline{S}_{xu}$:

$$\overline{\Sigma} \ : \ x^+ = Ax + Bu, \tag{5.1}$$

$$\overline{S}_{xu} = S_{xu} - \overline{W}_\infty \times \{0\}, \tag{5.2}$$

where A and B are the same as in (2.1). Let $\overline{\mathcal{C}}_{\max}$ be the Maximal CIS of the nominal system $\overline{\Sigma}$ within $\overline{S}_{xu}$ and define:

$$\begin{aligned} \mathcal{C}_{outer,\nu} = \Big\{ x \in \mathbb{R}^n \ | \exists \{u_i\}_{i=0}^{\nu-1} \in \mathbb{R}^{m\nu}, \\ \big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu}, t = 0, \dots, \nu-1, \\ \mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{\nu-1}\big) \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty \Big\}. \end{aligned} \tag{5.3}$$

Proposition 5.0.1. *$\mathcal{C}_{outer,\nu}$ is an RCIS of Σ within S_{xu}.*

Proof. In this proof, we use the order cancellation lemma, a special case of [GU19, Thm. 4].

Lemma 5.0.2. *Let $X, Y \subset \mathbb{R}^n$ be two closed convex sets with Y bounded. A point $x \in \mathbb{R}^n$ is in X if and only if $x + Y \subseteq X + Y$.*

To prove that $\mathcal{C}_{outer,\nu}$ is an RCIS, we show that for any $x_0 \in \mathcal{C}_{outer,\nu}$, there exists u such that $(x_0, u) \in S_{xu}$ and for all $w \in W$, $Ax_0 + Bu + Ew \in \mathcal{C}_{outer,\nu}$. By definition of $\mathcal{C}_{outer,\nu}$, there exists a sequence $\{u_i\}_{i=0}^{\nu-1}$ that, along with x_0, satisfies the conditions in (5.3). We aim to show that u_0 in $\{u_i\}_{i=0}^{\nu-1}$ is a feasible choice for u. Given (5.3), the reachable set from x_0 at time ν is:

$$\mathcal{R}_\Sigma\big(x_0, \{u_i\}_{i=0}^{\nu-1}\big) = \sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i + \overline{W}_\infty \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty,$$

with $\overline{W}_\infty$ and $\overline{\mathcal{C}}_{\max}$ being convex and $\overline{W}_\infty$ being bounded. By Lemma 5.0.2 we have that $\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i \in \overline{\mathcal{C}}_{\max}$. Since $\overline{\mathcal{C}}_{\max}$ is controlled invariant within $\overline{S}_{xu}$ for the nominal DTLS $\overline{\Sigma}$, there exists u_ν such that:

$$\begin{aligned} &\left(\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i, u_\nu\right) \in \overline{S}_{xu}, \\ &A\left(\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i\right) + B u_\nu = \sum_{i=1}^{\nu} A^{\nu-1-i} B u_i \in \overline{\mathcal{C}}_{\max}. \end{aligned} \tag{5.4}$$

Consider any $w \in W$ and define $x_1 = Ax + Bu_0 + Ew$:

$$\mathcal{R}_\Sigma(x_1, \{u_i\}_{i=1}^{\nu}) = \sum_{i=1}^{\nu} A^{\nu-1-i} B u_i + \overline{W}_\infty \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty. \tag{5.5}$$

From (5.4) we have that:

$$(\mathcal{R}_\Sigma(x_1, \{u_i\}_{i=1}^{\nu-1}), u_\nu) \subseteq S_{xu}. \tag{5.6}$$

Finally, note that for $t = 0, \cdots, \nu - 2$, we have:

$$\big(\mathcal{R}_\Sigma(x_1, \{u_i\}_{i=1}^{t}), u_{t+1}\big) \subseteq \big(\mathcal{R}_\Sigma(x_0, \{u_i\}_{i=0}^{t}), u_{t+1}\big) \subseteq S_{xu}. \tag{5.7}$$

From (5.5), (5.6), and (5.7) we verify that $x_1 \in \mathcal{C}_{outer,\nu}$. Thus, $\mathcal{C}_{outer,\nu}$ is an RCIS. □

The following theorem shows that $\mathcal{C}_{outer,\nu}$ is an outer bound of the projection of the proposed implicit RCIS.

Theorem 5.0.3 (Outer bound on $\pi_n(\mathcal{C}_{xv})$)**.** *For a companion system Σ_{xv} as in* (3.3)*, with arbitrary matrices P and H, let $\mathcal{C}_{xv}$ be an RPIS of Σ_{xv} within the companion safe set S_{xv}. The RCIS $\pi_n(\mathcal{C}_{xv})$ is a subset of $\mathcal{C}_{outer,\nu}$, that is $\pi_n(\mathcal{C}_{xv}) \subseteq \mathcal{C}_{outer,\nu}$.*

Proof. Let $x \in \pi_n(\mathcal{C}_{xv})$. We show that $x \in \mathcal{C}_{outer,\nu}$. By definition of $\mathcal{C}_{xv}$, there exists a vector v such that:

$$\left(\mathcal{R}_\Sigma\left(x, \left\{HP^i v\right\}_{i=0}^{t-1}\right), HP^t v\right) \subseteq S_{xu}, \text{ for all } t \geq 0. \tag{5.8}$$

Define $u_t = HP^t v$. We want to verify that x and $\{u_i\}_{i=0}^{\nu-1}$ satisfy the two conditions in the definition of (5.3). The first condition is immediately satisfied by (5.8). It is left to show that $\mathcal{R}_\Sigma(x, \{u_i\}_{i=0}^{\nu-1}) \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty$. That is:

$$\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i + \overline{W}_\infty \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty.$$

By Lemma 5.0.2, it is equivalent to prove that:

$$\overline{x} \equiv \sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_i \in \overline{\mathcal{C}}_{\max}.$$

By (5.8), we have that for $t \geq 0$:

$$\begin{aligned}\left(\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_{i+t} + \overline{W}_\infty, u_{v+t}\right) \subseteq S_{xu} &\Leftrightarrow \left(\sum_{i=0}^{\nu-1} A^{\nu-1-i} B u_{i+t}, u_{\nu+t}\right) \in \overline{S}_{xu} \\ &\Leftrightarrow \left(\mathcal{R}_{\overline{\Sigma}}(\overline{x}, \{u_i\}_{i=\nu}^{\nu+t-1}), u_{\nu+t}\right) \in \overline{S}_{xu}\end{aligned} \tag{5.9}$$

According to (5.9), the control sequence $\{u_i\}_{i=\nu}^{\nu+t-1}$ guarantees that the trajectory of $\overline{\Sigma}$ starting at $\overline{x}$ stays within $\overline{S}_{xu}$ for all $t \geq 0$. Thus, $\overline{x}$ must belong to the Maximal CIS of Σ in $\overline{S}_{xu}$. That is, $\overline{x} \in \overline{\mathcal{C}}_{\max}$. □

Note here that the set $\mathcal{C}_{outer,\nu}$, which serves as an outer bound for the set computed by our method, is as hard to compute as the Maximal RCIS. Given Theorem 5.0.3 we have:

$$\pi_n\left(\mathcal{C}_{xv}\right) \subseteq \mathcal{C}_{outer,\nu} \subseteq \mathcal{C}_{\max}. \tag{5.10}$$

Thus, the projection of our implicit RCIS can coincide with the Maximal RCIS, for appropriately selected matrices P and H, only if $\mathcal{C}_{outer,\nu} = \mathcal{C}_{\max}$ in (5.10). This potential gap between our approximation and the Maximal RCIS is due to the fact that our method uses open-loop forward reachability constraints under distrurbances. Finally, the following theorem establishes weak completeness of our method.

Theorem 5.0.4 (Weak completenss)**.** *The set $\mathcal{C}_{outer,\nu}$ is nonempty, if and only if, there exist matrices P and H such that the corresponding implicit RCIS $\mathcal{C}_{xv}$ is nonempty. Specifically, $\mathcal{C}_{outer,\nu}$ is nonempty, if and only if, $\mathcal{C}_{xv,(0,1)}$ is nonempty.*

Proof. We want to show that $\mathcal{C}_{outer,\nu}$ is nonempty if and only if $\mathcal{C}_{xv,(0,1)}$ is nonempty, where $\mathcal{C}_{xv,(0,1)}$ is defined in (4.4) with respect to system Σ and safe set S_{xu}.

Since $\pi_n\left(\mathcal{C}_{xv,(0,1)}\right) \subseteq \mathcal{C}_{outer,\nu}$, immediately nonemptyness of $\mathcal{C}_{xv,(0,1)}$ implies nonemptyness of $\mathcal{C}_{outer,\nu}$.

To show the opposite direction, suppose that $\mathcal{C}_{outer,\nu}$ is nonempty. Then $\overline{\mathcal{C}}_{\max}$ is nonempty. Using similar arguments as in the proof of [AT19, Thm. 3.3], we know that $\overline{\mathcal{C}}_{\max}$ is nonempty, if and only if, there exists a fixed point $x \in \overline{\mathcal{C}}_{\max}$ along with a u such that $(x,u) \in \overline{S}_{xu}$ and $Ax + Bu = x$. Also, note that $A\overline{W}_\infty + EW = \overline{W}_\infty$. Thus, we have:

$$\begin{aligned} &(x + \overline{W}_\infty, u) \in S_{xu}, \\ &A(x + \overline{W}_\infty) + Bu + EW = x + \overline{W}_\infty. \end{aligned} \tag{5.11}$$

According to (5.11), for any $y \in x + \overline{W}_\infty$, we have $(y,u) \in S_{xu}$ and $Ay + Bu + EW \subseteq x + \overline{W}_\infty$, which implies that $x + \overline{W}_\infty$ is an RCIS of Σ within S_{xu}. By the definition of $\mathcal{C}_{xv,(0,1)}$, it is easy to check that $(x + \overline{W}_\infty, u) \subseteq \mathcal{C}_{xv,(0,1)}$. Thus, $\mathcal{C}_{xv,(0,1)}$ is nonempty. □

Corollary 5.0.5 (Completeness in the absence of disturbances)**.** *In the absence of disturbances, $\mathcal{C}_{outer,\nu} = \mathcal{C}_{max}$. Thus, by Theorem 5.0.4, there exists P and H such that $\mathcal{C}_{xv}$ is nonempty, if and only if, $\mathcal{C}_{max}$ is nonempty. That is, the proposed method is complete.*

The significance of Theorem 5.0.4 lies in allowing to quickly check nonemptiness of $\mathcal{C}_{outer,\nu}$ by computing $\mathcal{C}_{xv,(0,1)}$, which we can do in closed-form.

Even though the gap between $\mathcal{C}_{outer,\nu}$ and $\mathcal{C}_{\max}$ is still an open question at the writing of this manuscript, we show that $\pi_n(\mathcal{C}_{xv})$ can actually converge to its outer bound for a specific choice of H and P matrices.

Theorem 5.0.6 (Convergence to $\mathcal{C}_{outer,\nu}$)**.** *There exist matrices H and P such that the projection of the corresponding implicit RCIS $\mathcal{C}_{xv}$ approaches $\mathcal{C}_{outer,\nu}$. Specifically, if H and P are as in (4.1), by increasing τ in (4.1), $\pi_n(\mathcal{C}_{xv})$ converges to $\mathcal{C}_{outer,\nu}$ exponentially fast.*

Proof. Recall from (5.3) that $\mathcal{C}_{outer,\nu}$ is:

$$\begin{aligned}\mathcal{C}_{outer,\nu} = \Big\{ x \in \mathbb{R}^n \mid & \exists \{u_i\}_{i=0}^{\nu-1} \in \mathbb{R}^{m\nu}, \\ & \big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu}, t = 0, \dots, \nu-1, \\ & \mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{\nu-1}\big) \subseteq \overline{\mathcal{C}}_{\max} + \overline{W}_\infty \Big\}.\end{aligned}$$

Using the matrices H and P as in (4.1), by the definition of $\mathcal{C}_{xv,(\tau,\lambda)}$, we can write that $\pi_n(\mathcal{C}_{xv})$ is:

$$\begin{aligned}\pi_n(\mathcal{C}_{xv}) = \Big\{ x \in \mathbb{R}^n \mid & \exists v \in \mathbb{R}^{(\tau+\lambda)m} \ s.t. \ u_t = HP^t v, t \geq 0, \\ & \big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu}, t = 0, \dots, \nu-1, \\ & \big(\mathcal{R}_\Sigma\big(x, \{u_i\}_{i=0}^{t-1}\big), u_t\big) \subseteq S_{xu}, t = \nu, \dots, \nu+\tau+\lambda-1 \Big\}.\end{aligned}$$

Since H and P are as in (4.1), it is actually the case that $u_t = v(t)$, $t = 0, \dots, \nu+\tau+\lambda-1$, where $v(t) \in \mathbb{R}^m$ is the t-th element of $v \in \mathbb{R}^{(\tau+\lambda)m}$.

We want to show that $\pi_n(\mathcal{C}_{xv})$ converges to $\mathcal{C}_{outer,\nu}$ for some value of τ. First, by selecting $\tau \geq \nu$, it is straightforward that the constraints for $t = 0, \dots, \nu-1$ are the same in both sets above.

Next, given the expression for the reachable set in (2.5), the following set represents the last constraint in $\pi_n(\mathcal{C}_{xv})$:

$$\begin{aligned}\overline{\mathcal{C}}_{\max,(\tau,\lambda)} = \Big\{ x \in \mathbb{R}^n \mid \exists v \in \mathbb{R}^{(\tau+\lambda)m} \\ s.t.\ u_t = HP^t v, t \geq 0,\ x = \sum_{i=1}^{\nu} A^{i-1} B u_{t-i},\ t = \nu \\ \left(\sum_{i=1}^{\nu} A^{i-1} B u_{t-i}, u_t \right) \in \overline{S}_{xu},\ t = \nu, \ldots, \nu+\tau-1 \qquad (A) \\ \left(\sum_{i=1}^{\nu} A^{i-1} B u_{t-i}, u_t \right) \in \overline{S}_{xu},\ t = \nu+\tau, \ldots, \nu+\tau+\lambda-1 \ (B) \Big\},\end{aligned}$$

where $\overline{S}_{xu} = S_{xu} - \overline{W}_\infty \times \{0\}$. The constraint (B) in $\overline{\mathcal{C}}_{\max,(\tau,\lambda)}$ represents the set of points in $\overline{S}_{xu}$ rendered invariant by a periodic input with period λ, since for $t > \tau$ the input $u_t = HP^t v$ is already periodic. Its invariance is proven in [ALO21b]. This set naturally includes any fixed point of the nominal system $\overline{\Sigma}$ in $\overline{S}_{xu}$. Moreover, given the *controlled predecessor* of a set under system $\overline{\Sigma}$:

$$\mathrm{Pre}_{\overline{\Sigma}}(X) = \pi_n\left(\{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m | (Ax+Bu) \in X\}\right),$$

constraints (A) and (B) define the points that after τ steps are rendered periodically invariant. In other words, the set $\overline{\mathcal{C}}_{\max,(\tau,\lambda)}$ is the τ-step controlled predecessor within $\pi_n\left(\overline{S}_{xu}\right)$ of an invariant set that contains fixed points. Then, we can rewrite $\pi_n(\mathcal{C}_{xv})$ as:

$$\begin{aligned}\pi_n(\mathcal{C}_{xv}) = \Big\{ x \in \mathbb{R}^n \mid \exists v \in \mathbb{R}^{(\tau+\lambda)m}\ s.t.\ u_t = HP^t v, t \geq 0, \\ \left(\mathcal{R}_\Sigma\left(x, \{u_i\}_{i=0}^{t-1}\right), u_t\right) \subseteq S_{xu}, t = 0, \ldots, \nu-1, \\ \left(\mathcal{R}_\Sigma\left(x, \{u_i\}_{i=0}^{\nu-1}\right), u_t\right) \subseteq \overline{\mathcal{C}}_{\max,(\tau,\lambda)} + \overline{W}_\infty \Big\}.\end{aligned}$$

The work in [GC87, CG86] proves that by taking a finite number τ of $\mathrm{Pre}_{\overline{\Sigma}}$ steps (the number of steps is defined by A, B, $\overline{S}_{xu}$) we can exponentially converge to $\overline{\mathcal{C}}_{\max}$ starting from a set that contains a fixed point. This shows that for our method, by selecting a large enough τ, we have that $\overline{\mathcal{C}}_{\max,(\tau,\lambda)}$ converges to $\overline{\mathcal{C}}_{\max}$. Consequently, making the second set of constraints of $\mathcal{C}_{outer,\nu}$ and $\pi_n(\mathcal{C}_{xv})$ the same and concluding this proof. □

Corollary 5.0.7 (Convergence in the absence of disturbances)**.** *Similarly to the previous corollary, as in the absence of disturbances* $\mathcal{C}_{outer,\nu} = \mathcal{C}_{max}$*, by Theorem 5.0.6, the proposed method can converge to the Maximal CIS.*

CHAPTER 6

Implicit invariant sets in practice: controlled invariant sets in one move

Using the proposed results, one has the option to project the implicit RCIS back to the original space and obtain an explicit RCIS as proposed in the two-move approach [AT19, AT20, ALO21b]. However, the required projection from a higher dimensional space becomes the bottleneck of this approach.

One of the goals of this manuscript is to establish that in a number of key control problems explicit knowledge of the RCIS is not required and the implicit RCIS suffices. In particular, we study the problem of *supervision* in Section 6.1, the problem of *safe planning* in Section 6.2, and the problem of *determining safe hyper-boxes* in Section 6.3. For these problems, we show how the proposed methodology can be used *online* as the only requirement is the implicit RCIS which admits a closed-form expression. Later, in Section 7, we provide experiments based on the studies of this section.

6.1 Supervision of a nominal controller

In many scenarios, when synthesizing a controller for a plant, the objective is to meet a performance criterion while always satisfying a safety requirement. This gives rise to the problem of *supervision*.

Problem 1 (Supervisory Control)**.** *Consider a system Σ, a safe set S_{xu}, and a nominal controller designed to meet a performance objective. The supervisory control problem asks at*

each time step to evaluate if, given the current state x, the input $\tilde{u}$ chosen by the nominal controller keeps the next state of Σ in a RCIS in the safe set. If not, correct $\tilde{u}$ *by selecting an input that does so.*

To solve Problem 1 one has to guarantee *at every step* that the pairs of states and inputs respect the safe set S_{xu}. A natural way to do so is by using an RCIS. The supervision framework operates as follows. Given an RCIS $\mathcal{C}$, assume that the initial state of Σ lies in $\mathcal{C}$. The nominal controller provides an input $\tilde{u}$ to be executed by Σ:

- if $\tilde{u}$ is safe, that is $(x, \tilde{u}) \in S_{xu}$ and $\forall w \in W$, $Ax + B\tilde{u} + Ew \in \mathcal{C}$, then its execution is allowed;
- else $\tilde{u}$ is corrected by selecting an input u_{safe} such that $(x, u_{safe}) \in S_{xu}$ and $\forall w \in W$, $Ax + Bu_{safe} + Ew \in \mathcal{C}$. Existence of u_{safe} is guaranteed in any RCIS by Definition 3.

In practice an explicit RCIS is not needed. One can exploit any of the proposed implicit RCISs to perform supervision. Consider an implicit RCIS $\mathcal{C}_{xv}$ for Σ within S_{xu}, as in Theorem 3.3.3. Then supervision of an input $\tilde{u}$ is performed by solving the following optimization problem:

$$\begin{aligned} \min_{v} \ & ||Hv - \tilde{u}||_2^2 \\ \text{s.t.} \ & (x, Hv) \in S_{xu}, \\ & (Ax + BHv + Ew, Pv) \in \mathcal{C}_{xv}, \forall w \in W. \end{aligned} \tag{6.1}$$

Notice that the input to Σ is Hv. By solving optimization problem (6.1) we compute the *minimally intrusive safe input.*

Remark 6 (Implicit RCIS for supervision)**.** *The optimization problem* (6.1) *uses the more general implicit RCIS $\mathcal{C}_{xv}$ from Theorem 3.3.3. One can instead use any of the sets $\mathcal{C}_{xv,(\tau,\lambda)}$ for some (τ, λ) or even the set $\mathcal{C}_{xv\zeta,\mathrm{L}}$ in* (4.8)*, which would result in a mixed-integer program instead.*

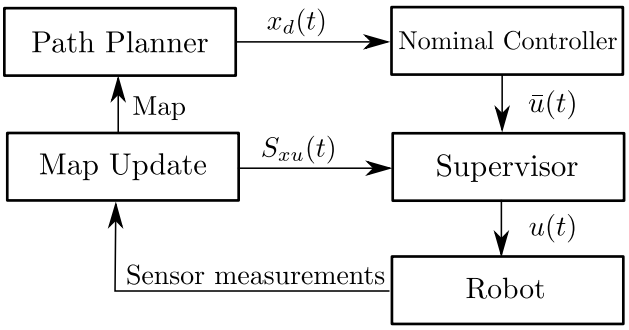


Figure 6.1: The overall safe online planning framework.

6.2 Safe online planning

Based on the discussed supervision framework, we utilize the proposed implicit RCIS to enforce safety constraints in online planning tasks. The goal here is to navigate a robot through unknown environments without collision with any obstacles. The map is initially unknown, and it is built and updated online based on sensor measurements, such as LiDAR. The robot must only operate in the detected obstacle-free region. To ensure this, given a path planning algorithm and a tracking controller, we supervise the controller inputs based on the implicit RCIS. The overall framework is shown in Figure 6.1.

The safe set for the robot imposes bounds on states and inputs, which do not change over time, but also constraints, e.g., on the robot's position, which are given by the obstacle-free region in the current map. As the detected obstacle-free region expands over time, the corresponding part of the safe set does as well. Thus, differently from Section 6.1, we have a time-varying safe set $S_{xu}(t)$ satisfying $S_{xu}(t) \subseteq S_{xu}(t+1)$, $t \geq 0$. As the implicit RCIS is constructed in closed-form, we can generate a new implicit RCIS $\mathcal{C}_{xv}(t)$ for each $S_{xu}(t)$. Then, at each time step t, for any $t' \leq t$, we supervise the nominal control input $\tilde{u}(t)$ by

solving the optimization problem:

$$\begin{aligned} \min_{v} \quad & ||Hv - \tilde{u}||_2^2 \\ \text{s.t.} \quad & (x, Hv) \in S_{xu} \\ & (Ax + BHv + Ew, Pv) \in \mathcal{C}_{xv}(t'), \forall w \in W. \end{aligned} \tag{6.2}$$

We denote the optimization problem in (6.2) by $\mathcal{P}(t, t')$. As $S_{xu}(t) \subseteq S_{xu}(t+1)$, $\mathcal{C}_{xv}(t')$ is a valid implicit RCIS in $S_{xu}(t)$ for all $t \geq t'$. Thus, as long as $\mathcal{P}(t, t')$ is feasible, the optimizer v^* of $\mathcal{P}(t, t')$ is a safe input that guarantees the next state lies in the RCIS. Furthermore, if $P(t, t')$ is feasible, by definition of RCIS, $P(t+1, t')$ is also feasible. Thus, if $P(0, 0)$ is feasible, for all $t > 0$, there exists $t' \leq t$ such that $P(t, t')$ is feasible. That is, the recursive feasibility of $P(t, t')$ is guaranteed. In practice, to take advantage of the latest map, we always select t' to be the latest time instant t^* for which $P(t, t^*)$ is feasible.

To summarize, at each time step, we first construct the implicit RCIS $\mathcal{C}_{xv}(t)$ based on the current map. Then, given the state and nominal control input, we solve $\mathcal{P}(t, t^*)$ to obtain the minimally intrusive safe input. This input guarantees that the state of the robot stays within $S_{xu}(t)$ for all $t \geq 0$, provided that $\mathcal{P}(0, 0)$ is feasible.

6.3 Safe hyper-boxes

For high dimensional systems, the exact representation of an RCIS $\mathcal{C}_x$ can be a set of thousands of linear inequalities. This provides reduced insight as it is quite difficult to clearly identify regions of each state vector entry that lie within the RCIS. In contrast, hyper-boxes are straightforward to grasp in any dimension, providing immediately information about which region of each state they contain. Based on this motivation, we explore how implicit RCISs can be used to find hyper-boxes that can be considered *safe* as in the following definition.

Definition 10 (Safe hyper-boxes)**.** *Consider a system Σ, a safe set S_x, and an RCIS*

$\mathcal{C}_x \subseteq S_x$. *Define a hyper-box* $\mathcal{B} = [\underline{b}_1, \overline{b}_1] \times \cdots \times [\underline{b}_n, \overline{b}_n] = [\underline{b}, \overline{b}] \subset \mathbb{R}^n$. *We call a hyper-box* $\mathcal{B}$ safe *if* $\mathcal{B} \subseteq \mathcal{C}_x$.

To simplify the presentation we only consider state constraints, S_x, instead of S_{xu}. Notice that by Definition 10, a safe hyper-box is not necessarily invariant. A safe hyper-box $\mathcal{B}$ entails the guarantee that the trajectory starting therein can remain in S_x forever, but not necessarily within $\mathcal{B}$. We aim to address the following problem.

Problem 2. *Find the largest*[1] safe *hyper-box* $\mathcal{B}$ *within* $\mathcal{C}_x$.

One can solve Problem 2, again, by use of an implicit RCIS $\mathcal{C}_{xv}$. A hyper-box $\mathcal{B}$ can be described by a pair of vectors $\left(\underline{b}, \overline{b}\right) \in \mathbb{R}^n \times \mathbb{R}^n$. Then, using similar arguments to Section 3, we compute in closed-form an implicit RCIS $\mathcal{C}_{\mathcal{B}}$ characterizing all hyper-boxes $\left(\underline{b}, \overline{b}\right)$ that remain in S_x under eventually periodic inputs. The eventually periodic inputs are given by a vector $v \in \mathbb{R}^{mL}$ with $L = \tau + \lambda$. Then, the set $\mathcal{C}_{\mathcal{B}}$ lives in $\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{mL}$ and is described by:

$$\begin{aligned} A^t \left[\underline{b}, \overline{b}\right] + \sum_{i=1}^{t} A^{i-1} B H P^{t-i} v \subseteq S_x - \sum_{i=1}^{t} A^{i-1} E W, \ t = 0, \ldots, \nu - 1, \\ \sum_{i=1}^{\nu} A^{i-1} B H P^{t-i} v \subseteq S_x - \sum_{i=1}^{\nu} A^{i-1} E W, \ t = \nu, \ldots, \nu + L - 1. \end{aligned} \tag{6.3}$$

The above constraints can all be written as linear inequalities in $\left(\underline{b}, \overline{b}, v\right) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{mL}$. Assume then that $\mathcal{C}_{\mathcal{B}}$ is:

$$\mathcal{C}_{\mathcal{B}} = \left\{ \left(\underline{b}, \overline{b}, v\right) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{mL} \;\middle|\; G_{\mathcal{B}} \begin{bmatrix} \underline{b} \\ \overline{b} \\ v \end{bmatrix} \leq f_{\mathcal{B}} \right\},$$

[1] *The largest, as measured by volume, hyper-box within a set might not be unique. We choose an appropriate heuristic for maximizing the volume of a set that leads to a well-defined convex optimization problem. Hence, the term "largest" refers to the heuristic used.*

where $G_{\mathcal{B}}$ and $f_{\mathcal{B}}$ are such that they model (6.3). One, then, solves Problem 2 using the implicit RCIS $\mathcal{C}_{\mathcal{B}}$ by solving the following convex optimization problem:

$$\begin{aligned} \max_{\eta=\left(\underline{b},\bar{b},v\right)} \ & \gamma\left(\bar{b}-\underline{b}\right) \\ \text{s.t.} \quad & G_{\mathcal{B}}\,\eta \leq f_{\mathcal{B}}, \end{aligned}$$

where $\gamma(y) = (\Pi_{i=1}^{n} y_i)^{\frac{1}{n}}$ is the geometric mean function, which is used as a heuristic for the volume of the hyper-box. Function γ is concave, and maximizing a concave function can be cast as a convex minimization problem [BV04].

Remark 7 (Invariant and recurrent hyper-boxes)**.** *Two special cases of the above are* invariant *hyper-boxes, when $\tau = 0$, $\lambda = 1$, see also [AT19], and* recurrent *hyper-boxes, when $\tau = 0$, $\lambda > 0$, see also [AT20, ALO21b].*

A related question to Problem 2 is to evaluate if a proposed hyper-box is safe. This is of interest when evaluating if the initial condition, e.g., the proposed hyper-box, of a problem guarantees generation of safe trajectories. Another scenario of interest consists in evaluating if a hyper-box around a point x_c, where the system is required to operate, is safe. Motivated by the above, consider the following problem.

Problem 3. *Given a configuration point $(x_c, \rho) \in \mathbb{R}^n \times \mathbb{R}^n_+$, evaluate if the hyper-box with center x_c and edge length 2ρ, i.e., $\mathcal{B} = [x_c - \rho, x_c + \rho] \subset \mathbb{R}^n$ is safe.*

Problem 3 is solved by a simple feasibility LP:

$$\begin{aligned} \text{find } & \eta = (x_c - \rho, x_c + \rho, v) \\ \text{s.t. } & G_{\mathcal{B}}\,\eta \leq f_{\mathcal{B}}. \end{aligned}$$

More complicated questions can be formulated. For instance, to find the largest safe box around a configuration point. The solutions of Problems 2 and 3 can be combined to solve this more elaborate question.

Remark 8 (Complexity when using implicit RCISs)**.** *In this section we showed how several key problems in control are solved without the need of projection and of an explicit RCIS, which results in extremely efficient computations since the implicit RCISs are computed in closed-form. The decision to be made is the size of the lift, i.e., the length of the input sequence. From a computational standpoint, this choice is only limited by how large an optimization problem one affords solving given the application.*

CHAPTER 7

Case studies

7.1 cis2m: a library for computing controlled invariant sets in 2 moves and closed-form implicit representations

To complement our theoretical contributions and make the results useful to the community, we have developed a library, named `cis2m`, for the computation of implicit and explicit RCISs based on the proposed method. At the time of writing this manuscript, `cis2m` comes as a standalone C++ library and also as a MATLAB package.

The case studies that follow in this chapter were performed on a MacBook Pro (Retina, Mid 2012) with a 2.3 GHz Quad-Core Intel Core i7 Processor and 8 GB 1600 MHz DDR3 RAM. Finally, instructions to replicate the case studies, can be found at the space where the library lives: `https://github.com/janis10/cis2m`.

7.2 Supervision of a quadrotor for obstacle avoidance using explicit RCIS

We begin by tackling the supervision problem, defined in Section 6.1, for the task of quadrotor obstacle avoidance. That is, we filter nominal inputs to the quadrotor to ensure collision-free trajectories. The dynamics of the quadrotor can be modeled as a non-linear system with 12 states [MD13]. Nonetheless, this system is differentially flat, which implies that the states and inputs can be rewritten as a function of the so-called flat outputs and a finite

number of their derivatives [ZS14]. Exploiting the differential flattness property, we obtain an equivalent linear system that expresses the motion of a quadrotor. Moreover, the original state and input constraints can be overconstrained by polytopes in the flat output space. For details on the above see [PAT21]. Then, the motion of a quadrotor can be described by:

$$x^+ = Ax + Bu + Ew,$$

where $A = \text{blkdiag}(A_1, A_2, A_3)$, $B = \text{blkdiag}(B_1, B_2, B_3)$, and:

$$A_i = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2!} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}, \; B_i = \begin{bmatrix} \frac{T_s^3}{3!} \\ \frac{T_s^2}{2!} \\ T_s \end{bmatrix}.$$

The state $x \in \mathbb{R}^9$ contains the 3-dimensional position, velocity, and acceleration, while the input $u \in \mathbb{R}^3$ is the 3-dimensional jerk. The matrix E and disturbance w are selected appropriately to account for various errors during the experiment.

The operating space for the quadrotor is a hyper-box with obstacles in $\mathbb{R}^3$, see Fig. 7.1. Therefore, the safe set is described as the obstacle-free space, a union of overlapping hyper-boxes in $\mathbb{R}^3$, along with box constraints on the velocity and the acceleration of the quadrotor:

$$S = \bigcup_{j=1}^{N} \left[\underline{p}_j, \overline{p}_j\right] \times [\underline{v}, \overline{v}] \times [\underline{a}, \overline{a}],$$

where $\left[\underline{p}_j, \overline{p}_j\right] \subset \mathbb{R}^3$, for $j = 1, \ldots, N$, is a hyper-box in the obstacle-free space, $[\underline{v}, \overline{v}] \subset \mathbb{R}^3$ denotes the velocity constraints, and $[\underline{a}, \overline{a}] \subset \mathbb{R}^3$ denotes the acceleration constraints. The safe set is a union of polytopes, while our framework is designed for convex polytopes. Since we already know the obstacle layout, we compute an explicit RCIS for each polytope in the safe set. As these polytopes overlap we expect, and it is actually the case in our experiments, that the RCISs do so as well. This allows, when performing supervision, to select the input that keeps the quadrotor into the RCIS of our choice when in the intersection of overlaping RCISs and, hence, navigate safely.

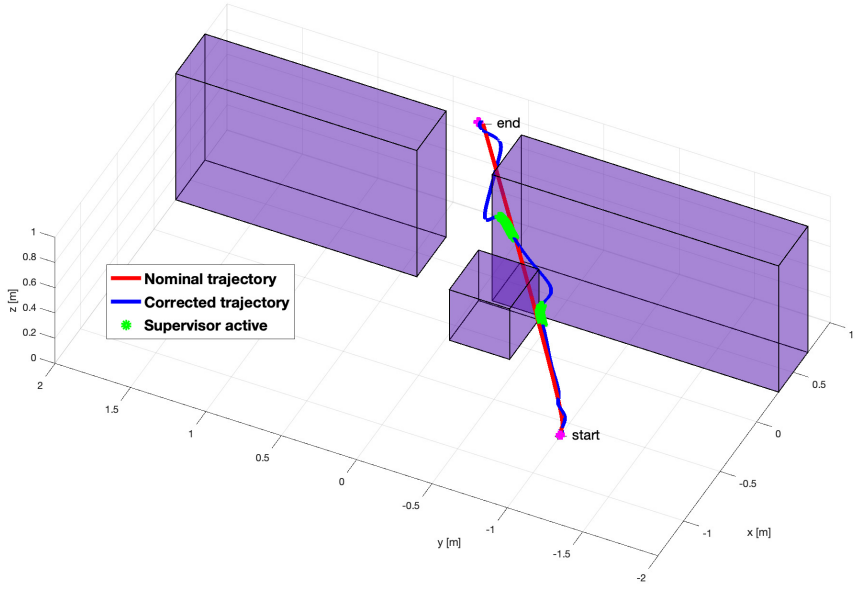
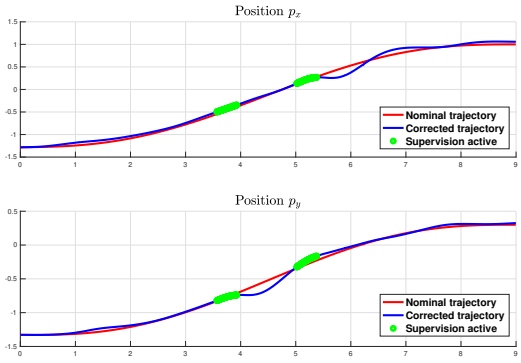
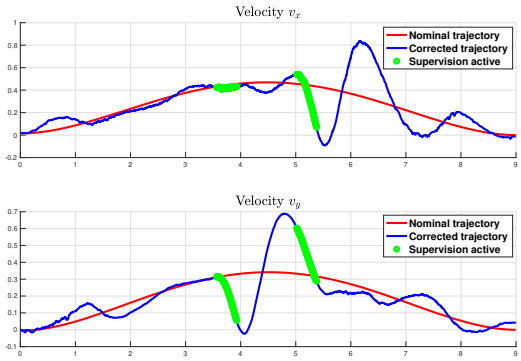


Figure 7.1: Quadrotor operational region. Obstacles in purple transparent boxes. Nominal trajectory (red), corrected trajectory (blue), supervision active (green).

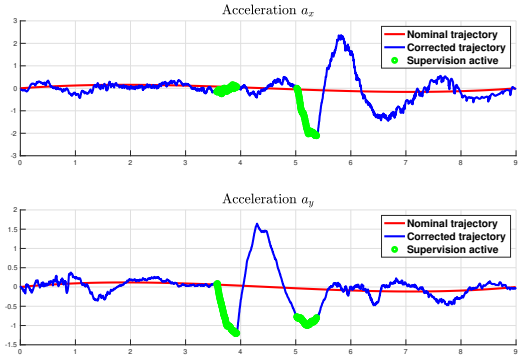
The goal of this case study is to ensure collision-free trajectory tracking. The nominal trajectory is the red straight line depicted in Fig. 7.1, moving the quadrotor from a start point to an end point through the obstacles. As we can appreciate, the supervised trajectory, blue curve, takes the quadrotor around the obstacles and, safely, to the end point. When the supervision is active, the quadrotor performs more aggressive maneuvers to avoid the obstacle as seen particularly in Fig. 7.2b and Fig. 7.2c, where we omit the z-axis as in this experiment the quadrotor maintains a relatively constant altitude. A video of the experiment is found at `https://tinyurl.com/drone-supervision-cis`. For visualizing the trajectory and the obstacles in the video, we used the Augmented Reality Edge Networking Architecture (ARENA) [CON].



(a) Quadrotor position



(b) Quadrotor velocity



(c) Quadrotor acceleration

Figure 7.2: Quadrotor trajectory in x-y plane: nominal trajectory (red), corrected trajectory (blue), supervision active (green).

In this experiment we utilized the explicit RCIS $\mathcal{C}_x = \pi_n\left(\mathcal{C}_{xv,(\tau,\lambda)}\right)$ with $(\tau, \lambda) = (0, 6)$. Even though we computed the projection, this operation was done in just several seconds for this specific system. Further, as we see later in Section 7.4, our one-time projection scales better than the classical algorithm that computes the Maximal CIS. Notice that, as mentioned in Remark 5, we used just one of the individual implicit RCISs comprising this hierarchy level. Our hardware platform is the open-source Crazyflie 2.0 quadrotor. The position is measured in m, the velocity in m/s, the acceleration in m/s^2, and the jerk in m/s^3. The operating space for the position is $[-2, 2] \times [-2, 2] \times [0, 1]$ and the obstacles are as seen in Fig. 7.1. The starting point is $(-1.35, -1.30, 0.7)$ and the end point is $(1.0, 0.3, 0.7)$. The velocity constraints are $[-1.0, 1.0]$, the acceleration constraints are $[-2.83, 2.83]$, and the jerk constraints are $[-59.3, 59.3]$. The sampling time for the supervision was set at $T_s = 0.18s$. For the estimation of the state we use a Kalman filter where the measurements are the position of the quadrotor and its attitude as obtained by the motion capture system OptiTrack. The optimization problems were solved by GUROBI [Gur20].

7.3 Safe online planning using implicit RCIS

Next, we solve the safe online planning problem, discussed in Section 6.2, for ground robot navigation. The map is initially unknown and is built online based on LiDAR measurements. While navigating the robot needs to avoid the obstacles, indicated by the dark area in Fig. 7.3, and reach the target point. This case study is inspired by the robot navigation problem in [BBB19]. The robot's motion, using forward Euler discretization, is:

$$x^+ = \begin{bmatrix} \mathbb{I} & \mathbb{I}T_s \\ 0 & \mathbb{I} \end{bmatrix} x + \begin{bmatrix} 0 \\ \mathbb{I}T_s \end{bmatrix} u,$$

where the state $x = (p_x, p_y, v_x, v_y) \in \mathbb{R}^4$ is the robot's position and velocity on the 2D map, and the input $u = (u_1, u_2) \in \mathbb{R}^2$ is the acceleration.

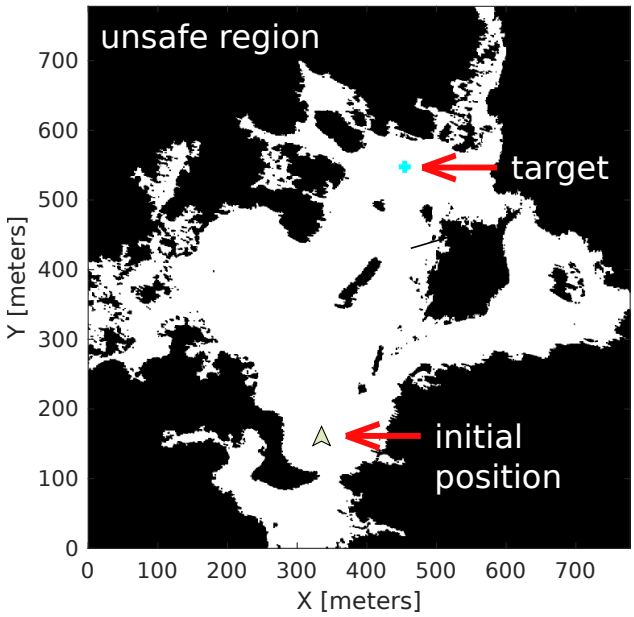


Figure 7.3: Robot operational space: initial position (yellow arrowhead), target position (cyan), unsafe region (dark area).

The safe set consists of two parts:

1. The time-invariant constraints $v_x, v_y \in [-\overline{v}, \overline{v}]$ and $u_1, u_2 \in [-\overline{u}, \overline{u}]$.

2. The time-varying constraint of (p_x, p_y) within the obstacle-free region, shown by the white nonconvex area in Fig. 7.3.

The obstacle-free region, denoted by $M(t) \subseteq R^2$, is determined by a LiDAR sensor using data up to time t. Combining the two constraints, the safe set at time t is:

$$S_{xu}(t) = \{(p_x, p_y, v_x, v_y, u_1, u_2) \mid (p_x, p_y) \in M(t),\ v_x, v_y \in [-\overline{v}, \overline{v}],\ u_1, u_2 \in [-\overline{u}, \overline{u}]\}.$$

Note that $M(t) \subseteq M(t+1)$ and, thus, $S_{xu}(t) \subseteq S_{xu}(t+1)$ for all $t \geq 0$.

The overall control framework was shown in Fig. 6.1. Initially, the map is blank and the path planner generates a reference trajectory assuming no obstacles. At each time instant t, the map is updated based on the latest LiDAR measurements and the path planner checks if

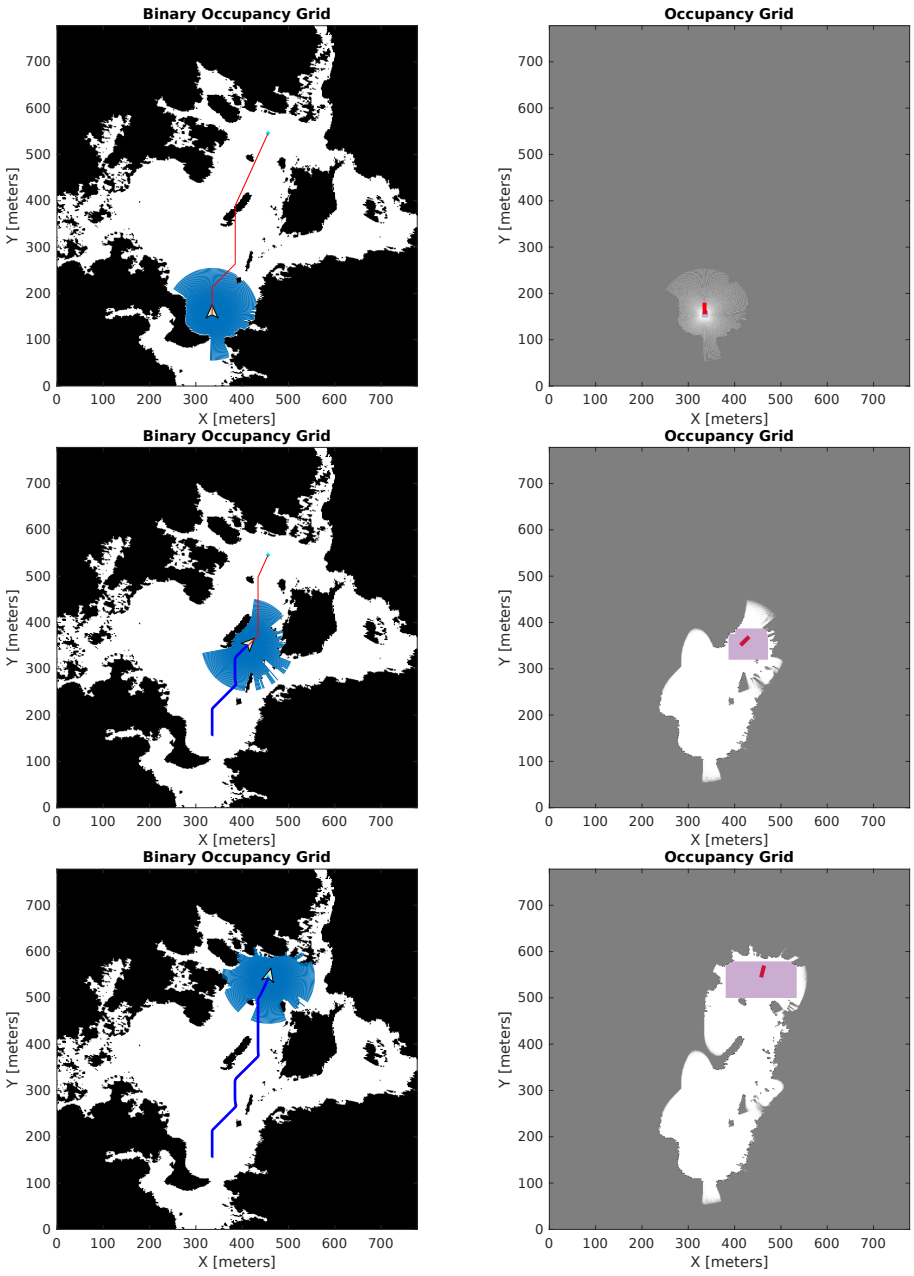


Figure 7.4: Simulation at $t = 0s$ (top), $40s$ (middle), and 78.6s (bottom). Left: reference path (red), actual trajectory (blue), LiDAR measurements (blue disks). Right: obstacle-free region $M(t)$ (white) and unknown region (grey); purple boxes are the 10 largest boxes in $M(t)$ that contain the current robot position.

the reference trajectory collides with any obstacles in the updated map. If so, it generates a new, collision-free, reference path. Then, the nominal controller provides a candidate input $\tilde{u} = (\tilde{u}_1(t), \tilde{u}_2(t))$ tracking the reference path. When updating the reference trajectory, a transient period is needed for the robot to follow the new reference tightly. Moreover, the path planner cannot guarantee satisfaction of the input constraints. To resolve these issues, we add a supervisory control to the candidate inputs. Based on the updated obstacle-free region $M(t)$, we construct the safe set $S_{xu}(t)$ and compute an implicit CIS $\mathcal{C}_{xv,(\tau,\lambda)}(t)$ within $S_{xu}(t)$. To handle the nonconvexity of $S_{xu}(t)$, we first compute a convex composition of $S_{xu}(t)$. Then, when constructing the implicit CIS, we let the reachable set at each time belong to one of the convex components in $S_{xu}(t)$, encoded by mixed-integer linear inequalities. For details see [LO21]. The convex decomposition of $S_{xu}(t)$ becomes more complex over time, which slows down the algorithm. To lighten the computational burden, we replace the full convex composition by the union of the 10 largest hyper-boxes in $S_{xu}(t)$ as the safe set. Given the constructed implicit CIS $\mathcal{C}_{xv,(\tau,\lambda)}(t)$ at time t, we supervise the nominal control input $\tilde{u}(t)$ by solving $\mathcal{P}(t, t^*)$ in (6.2), as discussed in Section 6.2. Note that $P(t, t^*)$ becomes a mixed-integer program as we introduced binary variables for the convex composition of the safe set and, therefore, in the implicit CIS.

For conducting our simulations, we use a linear feedback controller as the nominal controller. The MATLAB Navigation Toolbox is used to simulate a LiDAR sensor with sensing range of 100 m, update the map, and generate the reference path based on the A* algorithm. The simulation parameters are $(\tau, \lambda) = (6, 4)$, $T_s = 0.1s$, $\overline{v} = 5m/s$, $\overline{u} = 5m/s^2$. The mixed-integer program $\mathcal{P}(t, t^*)$ is implemented via YALMIP [Lof04] and solved by GUROBI [Gur20]. The average computation time for constructing the lifted set $\mathcal{C}_{xv,(\tau,\lambda)}(t)$ and solving $\mathcal{P}(t, t^*)$ at each time step is $2.87s$. The average computation time shows the efficiency of our method, considering the safe set is nonconvex and being updated at every time step.

The simulation results are shown in Fig. 7.4. The robot reaches the target region at $t = 78.6s$, and thanks to the supervisor, it satisfies the input and velocity constraints, while

always staying within the time-varying safe region. As a comparison, when the supervisor is disabled, the velocity constraint is violated at time $t = 1.2s$. The full simulation video can be found at `https://youtu.be/mB9ir0R9bzM` .

7.4 Scalability and quality

In this subsection we illustrate the scalability of the proposed method and compare with other methods in the literature. We consider a system of dimension n as in (2.1) that is already in Brunovsky normal form [Bru70].

$$A_n = \begin{bmatrix} 0 & \mathbb{I} \\ 0 & 0 \end{bmatrix}, \; B_n = \begin{bmatrix} \mathbb{0} \\ 1 \end{bmatrix},$$

where $A_n \in \mathbb{R}^{n \times n}$ and $B_n \in \mathbb{R}^n$. This assumption does not affect empirical performance measurements as the transformation that brings a system in the above form is system-dependent and, thus, can be computed offline just once. To generalize the assessment of performance, we generate the safe set as a random polytope of dimension n and we average the results over multiple runs. Moreover, we constraint our input to $[-0.5, 0.5]$ and the disturbance to $[-0.1, 0.1]$.

7.4.1 Scalability of implicit invariant sets

We first show the scalability of computing implicit invariant sets, both by computing the full hierarchy, i.e., all the invariant sets that form a hierarchy level, and by computing individual invariant sets themselves.

We begin with the case of no disturbances, that is, we compute implicit CISs. Under this scenario, Fig. 7.5 shows the runtimes to compute implicit CISs as the system dimension, n, grows. In particular, Fig. 7.5a and Fig. 7.5b show the computation times for $\mathcal{C}_{xv,\mathrm{L}}$, for different levels L of the hierarchy and for polytopes with $2n$ and n^2 constraints respectively. Our method computes implicit CISs in less than 0.5 seconds for systems of size $n = 200$

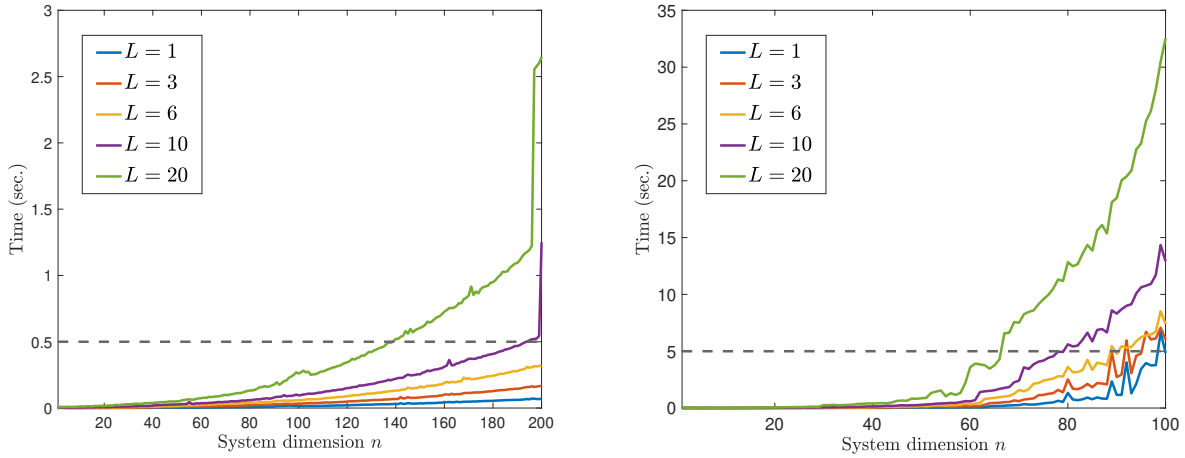
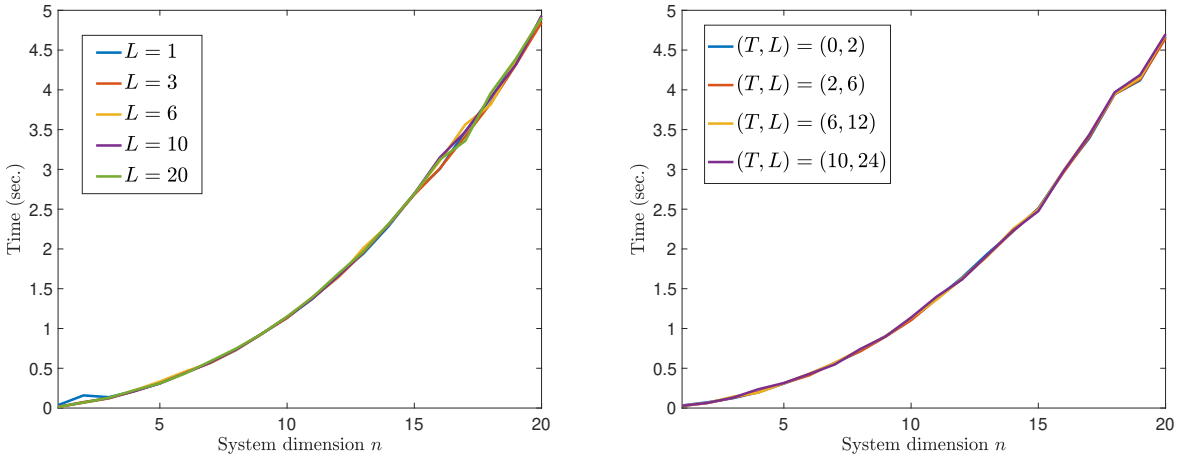


Figure 7.5: Absence of disturbances. Computation times for Implicit CISs for different levels L of the full hierarchy, i.e., computing L Implicit CISs per level. (a) Safe sets with $2n$ constraints, $n \leq 200$. (b) Safe sets with n^2 constraints, $n \leq 100$.

when the safe set has $2n$ constraints, and in around 5 seconds for $n = 100$ and safe sets with n^2 constraints, that is 10000 constraints in this example. This study validates the efficiency of our method in the case of nominal systems with no system model disturbances.

We now proceed to the case where system disturbances are present. This scenario is presented in Fig. 7.6. More particularly, in Fig. 7.6a and Fig. 7.6b, we observe that in the presence of disturbances computations are slower and, actually, are almost identical for different values of L. This is attributed to the presence of the Minkowsky difference in the closed-form expression (3.11) that dominates the runtime and depends on the nilpotency index of the system. Still, we are able to compute implicit RCISs in closed-form for systems with up to 20 states fairly efficiently in this experiment.

The above results suggest the applicability of our approach to scenarios involving online computations, as shown already in Section 7.3, and showcases the power of the implicit representation. Furthermore, in our experience, the numerical result of a projection operation,



(a) (b)

Figure 7.6: Presence of disturbances. Safe sets with $2n$ constraints, $n \leq 20$. Computation times for Implicit RCISs. (a) Different levels L of the hierarchy. (d) Individual implicit RCIS, $\mathcal{C}_{xv,(\tau,\lambda)}$, for different values of (τ, λ).

depending the method used, can be sometimes unreliable. Contrary to this, our closed-form expression does not suffer from such drawback.

7.4.2 Quality of the computed sets and comparison to other methods

We now compare our method with different methods in the literature, both in runtime and quality of the computed sets as measured by the percentage of their volume compared to the Maximal (R)CIS. Even though, we already provided a comprehensive analysis in terms of runtime for our method, we still present a few cases for the shake of comparison. We compare our approach to the Multi-Parametric Toolbox (MPT3) [HKJ13] that computes the Maximal (R)CIS, $\mathcal{C}_{max}$, the iterative approach in [TJ15] that computes low-complexity (R)CISs, and the one in [LTJ18] that computes ellipsoidal CISs.

The runtimes of each method are reported in Fig. 7.7. The difficulty of computing $\mathcal{C}_{max}$ is apparent from the steep corresponding curve. The low-complexity methods in [TJ15] and

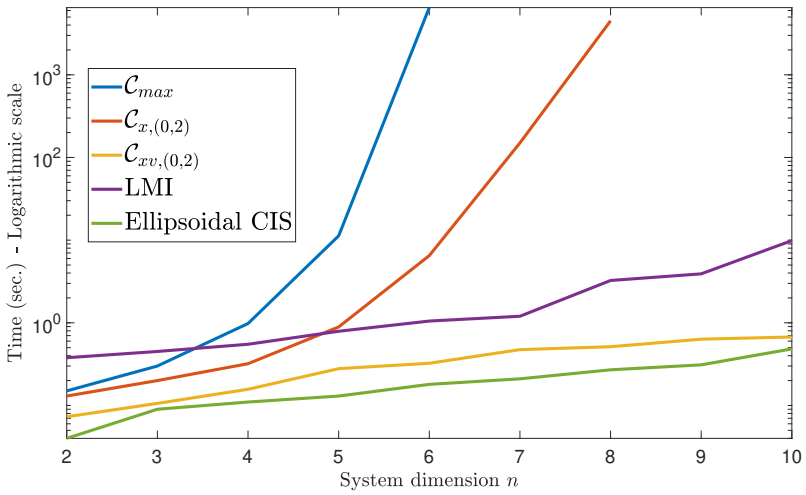


Figure 7.7: Computation times for $\mathcal{C}_{xv,(0,2)}$, its projection $\mathcal{C}_{x,(0,2)}$, the methods in [TJ15] and [LTJ18], and $\mathcal{C}_{max}$. Logarithmic scale. Note: [LTJ18] is evaluated in the absence of disturbances as it considers only nominal systems. For the other methods the performance without disturbance is similar or better.

[LTJ18] are considerably faster, and [LTJ18] that solves Sum-Of-Square Programs is slightly faster than even our implicit representation. However, our sets are superior in quality as we detail next.

First, in the absence of disturbances, the relative volume of the computed sets with respect to $\mathcal{C}_{max}$ is presented in Table 7.1. Since for $n \geq 7$ MPT3 does not terminate after several hours and the computed set before termination is not invariant, we present the relative volumes only for $2 \leq n \leq 6$. Our method returns a very close approximation of $\mathcal{C}_{max}$ even with small values of (τ, λ) and computes substantially larger sets compared to the other techniques. In other words, our implicit representation retains the best out of two worlds: computational efficiency and close approximations of $\mathcal{C}_{max}$. This is illustrated in two dimensions in Fig. 7.8.

In the presence of disturbances, the results are similar and are reported in Table 7.2, where we omit the method in [LTJ18] that was not designed for the presence of disturbances.

Table 7.1: Absence of disturbances. Volume percentage with respect to the Maximal CIS. Algorithms: Our method computing different implicit CISs $\mathcal{C}_{xv,(\tau,\lambda)}$, the LMI method in [TJ15], and the method in [LTJ18] computing ellipsoidal CISs. (S) denotes a singleton set.

	Our method			LMI method [TJ15]	Ellipsoidal CIS method [LTJ18]
System dimension	$\mathcal{C}_{xv,(0,2)}$	$\mathcal{C}_{xv,(2,2)}$	$\mathcal{C}_{xv,(4,2)}$		
$n=2$	100	100	100	42.43	45.69
$n=3$	100	100	100	16.31	24.66
$n=4$	99.92	100	100	3.69	14.41
$n=5$	99.75	100	100	0.47	10.50
$n=6$	97.81	99.07	100	0 (S)	3.89

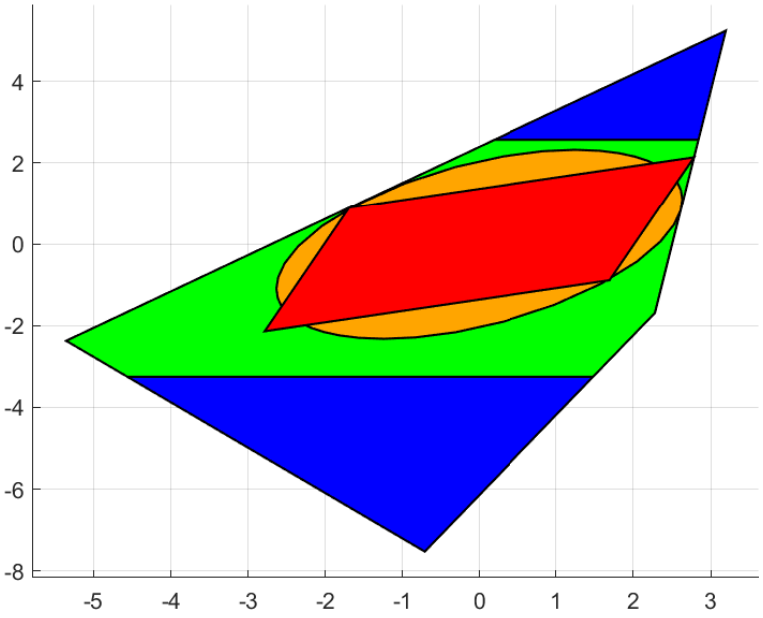


Figure 7.8: Two dimensional system in Brunovsky normal form. Randomly generated safe set (blue), $\mathcal{C}_{x,(1,2)} = \mathcal{C}_{max}$ (green), $\mathcal{C}_{ellips}$ from [LTJ18] (orange), and $\mathcal{C}_{lmi}$ from [TJ15] (red).

By increasing the magnitude of the disturbance, it is the case that some times our method could fail to compute non-empty sets. This is expected given the weak completeness result of Theorem 5.0.4. However, it is empirically the case even in the presence of disturbances,

Table 7.2: Presence of disturbances: volume percentage with respect to the Maximal RCIS. Algorithms: Our method computing different implicit RCISs $\mathcal{C}_{xv,(\tau,\lambda)}$ and the LMI method in [TJ15]. (S) denotes a singleton set.

Volume (%)	Our method			LMI method [TJ15]
System dimension	$\mathcal{C}_{xv,(0,2)}$	$\mathcal{C}_{xv,(2,2)}$	$\mathcal{C}_{xv,(4,2)}$	
$n=2$	100	100	100	31.99
$n=3$	98.24	99.67	99.96	16.35
$n=4$	99.02	99.42	99.88	4.36
$n=5$	98.75	99.74	99.81	3.64
$n=6$	91.17	96.07	97.91	0 (S)

Table 7.3: Computation time when projecting the implicit RCIS, $\mathcal{C}_{xv,(\tau,\lambda)}$, to obtain the explicit CIS $\mathcal{C}_{x,(\tau,\lambda)} = \pi_n\left(\mathcal{C}_{xv,(\tau,\lambda)}\right)$ for various values of (τ,λ). Times are in seconds.

System dimension	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$
$\mathcal{C}_{x,(0,2)}$	0.295	0.533	0.873	1.936	31.41	195.4	4970.7
$\mathcal{C}_{x,(2,2)}$	0.402	0.711	1.119	2.126	134.3	>7200	>7200
$\mathcal{C}_{x,(2,4)}$	0.541	0.890	1.359	2.553	603.4	>7200	>7200
$\mathcal{C}_{x,(3,8)}$	0.635	0.981	2.108	4.783	971.6	>7200	>7200
$\mathcal{C}_{max}$	0.369	0.895	2.410	21.62	6365.1	>7200	>7200

that when we compute non-empty sets they approximate $\mathcal{C}_{max}$ closely.

Finally, Table 7.3 presents the times to compute explicit RCISs for various pairs (τ,λ). Notice that the selected pairs (τ,λ) result in increasingly larger RCISs as outlined in Remark 5. Comparing the runtimes to obtain an explicit RCIS to the time needed to compute $\mathcal{C}_{max}$, we see that even when computing the explicit representation, our one-time projection outperforms the classical algorithm that requires smaller projections, but at each iteration. Of course, selecting a (τ,λ) that belongs to higher levels of the hierarchy increases the projection time due to the larger projection gap. As the reported times show, this becomes quite cumbersome when considering larger systems. However, our approach still provides an affordable way to compute explicit CISs by selecting lower levels of the hierarchy, e.g., $(\tau,\lambda)=(0,2)$.

CHAPTER 8

Discussion & Conclusion

The first part of the manuscript discussed a method for computing RCISs for discrete-time linear systems for which there exist a feedback transformation that makes the A matrix nilpotent. In particular, every controllable linear system belongs to this class. By considering controllers that exhibit eventually periodic behavior, a closed-form expression for an implicit RCIS, in the space of states and finite input sequences, is provided, as well as a hierarchy of RCISs for a special controller choice. The proposed method was validated in a number of safety-critical scenarios, including supervision for obstacle avoidance of a Crazyflie 2.0 quadrotor, and the scalability of the efficient implicit representation was established on high dimensional systems.

At this point we want to outline a number of potential open questions. Recall that we compute a closed-form expression for an implicit RCIS by using linear, eventually periodic controllers. A question that arises is whether there exist other controllers for linear systems that encapsulate the behavior of the proposed method and still provide similar computational performance. In other words, what is the maximal class of controllers we can consider in order to provide a closed-form expression for a polyhedral implicit RCIS given a polyhedral safe set and a linear system? Moreover, at the time of writing, another open question concerns the gap between the outer bound on the proposed method $\mathcal{C}_{outer}$ and the Maximal RCIS $\mathcal{C}_{max}$ in presence of disturbances. Ideally, we would like to be able to characterize the existence and the size of such gap given the linear system model, the safe set, and the disturbance set. When does such a gap exist and how close is $\mathcal{C}_{outer}$ to $\mathcal{C}_{max}$? Answering such a question would

allows us to further understand the limitations of the proposed method. From a practical perspective, it would be interesting to explore if one can obtain a minimal representation, i.e., irredundant in terms of inequality constraints, of the closed-form implicit RCIS. This would not only speed up the computation of the implicit RCIS, but it will also speed up its projection back to the original space, as well as, any of the optimization problems that involve the implicit RCIS as part of the constraints, e.g., the supervision framework we discussed.

Another interesting discussion concerns some limitations in the problem we consider. Our method considers polyhedral safe sets, that is state, input, and mixed state input constraints that are described by linear inequalities. Even though this covers already a large amount of applications, there are important scenarios with non-polyhedral or non-convex constraints. In such cases, one can still benefit from using our method by under-approximmating the given sets with the largest polyhedral set contained therein or even disjunctions of polyhedral sets. This introduces a trade-off: on the one hand obtaining a conservative RCIS for the given scenario and therefore a conservative set of safe actions, and on the other doing so very efficiently, in closed-form, and benefiting from the formal safety guarantees that it provides. Such a scenario was actually considered in Section 6.2 and 7.3 where the safe set obtained by our LiDAR sensor is highly non-convex and we operated on disjunctions of polyhedral sets. Notice, however, that operating on disjunctions of sets comes at the cost of solving mixed-integer programs instead of convex programs from an optimization perspective. In the same line of thinking, a case of interest is handling dynamic input constraints. For instance, our actuation is such that the available inputs at a given moment depend on the inputs used on previous time steps. Such a scenario can be incorporated in our static polyhedral constraints by augmenting the state of the system to include the appropriate sequence of inputs and impose constraints on this sequence. Based on the above, the proposed method can cover a wide variety of important scenarios.

Part II

rLTL verification: now faster than ever before!

CHAPTER 9

Introduction

As Cyber-Physical Systems (CPS) inevitably become increasingly complex, the ability to completely guarantee correctness of their design and implementation via exhaustive testing fades. Moreover, almost every aspect of contemporary life is becoming intertwined with CPSs, such smart grids, smart cities, mobility on demand and autonomous vehicles, and even medical devices. Consequently, in an attempt to reduce design errors, formal methods have been investigated to support modeling and verification of CPS and, in particular, of its reactive components.

Most work in formal methods has focused on system correctness, i.e., in ensuring that systems are guaranteed to meet their design specifications. We argue that correctness is necessary, but not sufficient for a good design when a reactive system interacts with an ever-changing uncontrolled environment. To illustrate this point, just consider the correctness specifications for open reactive systems, which are typically written in the form of an implication:

$$\varphi \Rightarrow \psi, \tag{9.1}$$

where φ is an environment assumption and ψ is a system guarantee. In Linear Temporal Logic (LTL) the implication in (9.1) is equivalent to $\neg\varphi \vee \psi$, and, ergo, whenever the assumption φ is violated the above specification yields no information on the guarantee. In other words, the system can behave arbitrarily. Thus, in addition to correctness, systems should also be designed to be robust, i.e., small deviations from the assumptions made at design time should lead to, at most, small violations of the design specifications. While it

is hard to dispute that design time assumptions may not hold in the environments where systems will actually be deployed, since these may not be completely known at design time or may be evolving over time, how to formally describe robustness is a question that has not received enough attention despite the recent efforts described in Section 9.1. In this part of the manuscript, we address this problem by studying a recently developed logic, termed robust Linear-time Temporal Logic and abbreviated as rLTL, that allows to specify robustness. Its syntax closely mirrors that of LTL to lower the barriers to its adoption. Its semantics, however, is different in many regards. In particular, it is a many-valued logic so that one can reason about the different ways in which assumptions and guarantees can be violated.

To shed more light into the mechanics of rLTL, consider the LTL formula $\Box p$ with p an atomic proposition. There is *only one way* in which this formula is satisfied, namely when p holds at every time step. In contrast, there are *several ways* in which this formula can be violated over an infinite trace: (1) the worst possible violation occurs when p fails to hold at every time step; (2) a slightly better scenario is where p holds for at most finitely many time instants; (3) better yet would be that p holds at infinitely many instants, while still failing to hold at infinitely many instants; (4) finally, among all the possible ways in which $\Box p$ can be violated, the most preferable case would be the one where p fails to hold for at most finitely many time instants. The semantics of rLTL is exactly designed to distinguish between these different ways.

In preliminary work [TN16], we introduced a fragment of rLTL that only contained the always and eventually operators. We showed, in that restricted context, that we can decide if a system satisfies an rLTL formula φ by using an automaton with size $\mathcal{O}\left(5^{|\varphi|}\right)$, where $|\varphi|$ denotes the length of φ. The corresponding automaton for LTL has size $\mathcal{O}\left(2^{|\varphi|}\right)$ and the change in the base of the exponential follows from the fact that LTL is a 2-valued logic, whereas rLTL is 5-valued. This manuscript offers a fragment of rLTL that also includes the next, until, and release operators, while placing a syntactic restriction on the antecedent

of nested implications. For the proposed fragment of rLTL the verification problem can be solved more efficiently, i.e., by using an automaton of size $\mathcal{O}\left(2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}\right)$, where $\kappa(\varphi)$ measures the number of unique subformulae of φ that contain always and release operators. Construction of smaller automata is achieved by using temporal testers and exploiting properties of the proposed fragment. In particular, this provides the upper bound $\mathcal{O}\left(3^{|\varphi|}\right)$ on the size of this automaton which is closer to the LTL bound. In prior work [APN18] we achieve the same complexity bound, but for a smaller fragment of rLTL. The fragment studied in [APN18] does not contain the release operator and allows at most one implication operator at the outermost level. Consequently, the fragment proposed here is substantially larger as evidenced by the second of our case studies, detailed next.

To illustrate the usefulness of rLTL and the proposed fragment, we offer several case studies that demonstrate: (1) how rLTL can identify a non-robust system, whereas LTL cannot, as it does not provide information about the guarantee of an implication when the environment deviates from the modeling assumptions; (2) how the proposed fragment contains the most important reactivity patterns [DAC99]; and (3) how the five truth values provide insightful information when a specification is violated, which can be useful for a designer seeking to improve the designed system and/or the specification. Moreover, the computational overhead associated with rLTL model-checking is relatively low with respect to LTL model-checking, and also rLTL model-checking, within the proposed fragment, scales similarly to LTL model-checking with respect to the size of the model-checked formula, as shown by our experiments. The content of this second part of the manuscript has been published in [APN22].

9.1 In a labyrinth of robustness

A number of efforts has been made in order to express the "correct" notion of robustness with regards to cyber-physical systems in formal methods. In this section, we present an

extensive, but not exhaustive, review of various formalizations of robustness. We begin by a series of approaches, which require the designer to provide information in addition to the desired specification.

In [BCG14], two quantitative robustness concepts are combined in a common framework for robust synthesis. The first one, robustness for safety, looks at how often the assumptions and the guarantees are violated, and asks for their ratio to be bounded by $k \in \mathbb{N}$ (k-robustness). Counting is achieved through error functions provided by the designer. The second concept, robustness for liveness, considers specifications of the form $\wedge_{i \in I} \Diamond \Box p_i \Rightarrow \wedge_{j \in J} \Diamond \Box q_j$, where p_i, q_j are atomic propositions, and then compares the number of violated assumptions to the number of violated guarantees. The rLTL semantics, even though being able to distinguish between the different ways in which a specification can be violated, does not distinguish between the violation of one assumption from the violation of multiple assumptions. Hence, the second approach cannot be compared to the one proposed here. Furthermore, we make no distinctions between safety and liveness properties.

Moreover, in [BCE19], a different framework for robust synthesis is proposed, that does not encompass the one above. Different notions of robustness are considered, e.g., a robust system satisfies a guarantee, even though a finite number, or even all, of the inputs are hidden/misread, or even though the assumption is violated finitely/infinitely often. Many of the considered notions are incorporated in rLTL, and in fact our definition of robustness allows systems to satisfy weaker guarantees, whenever the assumptions are also weakened, which is more general. Nonetheless, we cannot compare with the notions of robustness in [BCE19] that count the number of violations, since the rLTL semantics distinguishes only between zero, finite, and infinite violations of a specification.

In the intriguing work of [RBN16], a link between both MTL/LTL, and Linear Time-Invariant (LTI) filtering is established. Specifically, it is shown that LTI filtering corresponds to MTL if addition and multiplication are interpreted as max and min, and if true and false are interpreted as one and zero. By using different filtering kernels, one expresses weaker

or stronger interpretations of the same formula. However, this burdens the designer both with the choice of kernels and the use multiple semantics to reason about how weakening the assumptions leads to weakening the guarantees.

Contrary to all the approaches described so far, which require robustness metrics to be provided by the designer, when working with rLTL the designer only needs to provide the desired specification and no other information. Hence, we ease the designer's effort since it is not always clear which quantitative metric leads to the desired qualitative behavior.

Another interpretation of robustness is provided in [48] as the difference between the number of steps violating the guarantee and the number of steps violating the assumption of a reactive specification. A more robust reactive system produces smaller such differences, and robustness is evaluated quantitatively as a mean-payoff objective, averaged over all executions of the system. Alternatively, the discounted-sum can be used as a quantitative objective, which offers convergence properties over infinite runs. Moreover, this objective has been shown to pair well with qualitative LTL constraints, both in the reinforcement learning domain [WET15] and to obtain sound and efficient automata-based algorithms for quantitative reasoning [BCV21]. Again, such approaches are not comparable to ours since rLTL distinguishes only between zero, finite, and infinite violations of a specification.

In the domain of software systems, [ZGK20] defines robustness as the largest set of deviating environmental behaviors under which the system still guarantees a desired property. Therefore, robustness is defined, with respect to a property, as the set of all deviations under which a system continues to satisfy that property. Although this work focuses on computing robustness, rather than characterizing it, it is possible that certain temporal deviations could be expressed in rLTL. Additional noteworthy works, although incomparable with the methods described here, are [CGL10] and [MS09], which consider continuity properties of software expressed by the requirement that a deviation in a program's input causes a proportional deviation in its output. Although natural, these notions of robustness only apply to the Turing model of computation and not to the reactive model of computation employed

in this work.

A plethora of works exists regarding robustness of specifications when reasoning over real-valued, continuous-time signals, with the most prominent being [FP09], [DM10]. In these works, no discussion of the specific choices made when crafting the many-valued semantics is provided. Interestingly, though, the notion of "time robustness" in [DM10] is close to the one of rLTL in the sense that it measures the time needed for the truth value of a formula to change. Nevertheless, in this line of work robustness is derived from the real-valued nature of the signals, whereas in rLTL, we reason over the more classical setting of discrete-time and Boolean valued signals, with robustness derived from the temporal evolution of these signals. Consequently, the works of [FP09], [DM10], and their extensions can be considered of orthogonal and complementary nature to ours.

Another relevant approach of multi-valued extensions of LTL is found in [ABK16]. This work introduces two quantitative extensions of LTL, one by propositional quality operators termed LTL[$\mathcal{F}$], parameterized by a set $\mathcal{F}$ of functions over $[0,1]$, and one by discounting operators termed LTL$^{disc}[\mathcal{D}]$, parameterized by a set $\mathcal{D}$ of discounting functions. Both logics employ a many-valued variant of LTL to reason about quality, and the satisfaction value of a specification is a number in $[0,1]$, which describes the quality of the satisfaction. The use of a many-valued semantics in the context of quality is as natural as in the context of robustness. In fact, it was shown in [TN16] that by dualizing the semantics of rLTL in a specific sense we obtain a logic that is adequate to reason about quality. Nevertheless, there are strong conceptual differences between the approach taken in this work and the approach in [ABK16]. First, our notion of robustness or quality is intrinsic to the logic, while the approach in [ABK16] requires the designer to provide their own interpretation in the form of the sets $\mathcal{F}$ or $\mathcal{D}$ of functions that parameterize the logic. Second, there are several choices to define the logical connectives on the interval $[0,1]$. As an illustration for the latter, note that there are three commonly used conjunctions: Łukasiewicz's conjunction $a \wedge b = \max\{0, a+b-1\}$, Gödel's conjunction $a \wedge b = \min\{a,b\}$, and the product of real

numbers $a \wedge b = a \cdot b$ also known as Goguen's conjunction. Moreover, each such choice leads to a different notion of implication via residuation. Whether Gödel's conjunction, used in [ABK16], is the most adequate to formalize quality is a question not addressed in [ABK16]. In contrast, we carefully discuss and motivate all the choices made when defining the semantics of rLTL with robustness considerations.

The last body of work related to the contents of this manuscript is [KL07, AK14] on lattice automata and lattice LTL. The syntax of lattice LTL is similar to the syntax of LTL except that atomic propositions assume values on a finite lattice (which has to satisfy further restrictions such as being distributive). Although both lattice LTL as well as rLTL are many-valued logics, lattice LTL derives its many-valued character from the atomic propositions. In contrast, atomic propositions in rLTL are interpreted classically (i.e., they only assume two truth values). Therefore, the many-valued character of rLTL arises from the temporal evolution of the atomic propositions and not from the nature of the atomic propositions or their interpretation. In fact, if we only allow two truth values for the atomic propositions in lattice LTL, as is the case for rLTL, lattice LTL degenerates into LTL. Hence, these two logics capture orthogonal considerations, and results on lattice LTL and lattice automata do not shed light on how to address similar problems for rLTL.

9.2 Beyond rLTL verification

This second part of the manuscript studies the verification problem for rLTL. As we have already mentioned, the rLTL semantics allows for specifying robustness and is able to distinguish between the different ways in which a specification is violated in a qualitative manner, that is, between zero, finitely, and infinitely many violations. Nonetheless, rLTL does not distinguish between different numbers of assumptions being violated and, hence, cannot count over bounded time segments where a specification is defined as a conjunction of assumptions, each corresponding to a time step.

In addition to the verification problem, other problems for rLTL have been studied in the literature. The work in [TN16] studies the synthesis problem for a restricted fragment that contains only the always and eventually operators, while [TN15] extends the same results to full rLTL. Both works consider the environment to be antagonistic, which is not very realistic and leads to suboptimal controllers. This issue is addressed in [NNZ21] by introducing adaptive strategies, which are not more complex than the classical ones, and take advantage of the environment making bad moves. Furthermore, the runtime monitoring problem, i.e., checking properties of infinite words based on a given finite prefix, is studied in [MNS20] in the context of rLTL. Finally, different shortcomings of LTL, other than the lack of robustness, such as the limited expressiveness and the lack of quantitative features, have been addressed by other extensions like Linear Dynamic Logic [Var11] and Prompt-LTL [KPV09] respectively. While the above logics and rLTL address each shortcoming separately, the work in [NWZ19] shows how to combine any two of the aforementioned extensions and at the same time do not incur any additional complexity overhead from merging the logics.

CHAPTER 10

Review of Linear Temporal Logic (LTL) and LTL model-checking

Notation: Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers and $\mathbb{B} = \{0, 1\}$ the set of Boolean values with 0 interpreted as *false* and 1 interpreted as *true*. For a set A, let 2^A be the *powerset* of A, i.e., the set of all subsets of A, let A^ω be the set of all *infinite sequences* of elements of A, and let $\text{card}(A)$ denote the cardinality of A. An *alphabet* Σ is a finite, nonempty set whose elements are called *symbols*. An *infinite word* σ is an infinite sequence $\sigma = a_0 a_1 \cdots \in \Sigma^\omega$ of symbols with $a_i \in \Sigma$, $i \in \mathbb{N}$. For an infinite word $\sigma = \sigma_0 \sigma_1 \cdots \in \Sigma^\omega$ and $i \in \mathbb{N}$, let $\sigma(i) = \sigma_i$ denote the i-th symbol of σ and $\sigma_{i..}$ the (infinite) suffix of σ starting at position i, i.e., $\sigma_{i..} = \sigma_i \sigma_{i+1} \ldots \in \Sigma^\omega$. Notice that $\sigma_{0..} = \sigma$.

We begin by describing the syntax and semantics of *Linear Temporal Logic* (LTL) and recall the model-checking problem. This will form the backdrop against which rLTL will be introduced. The syntax of LTL is defined as follows.

Definition 11 (LTL syntax)**.** *Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. The set of all* LTL formulae *on $\mathcal{P}$, written* $\text{LTL}(\mathcal{P})$*, is the smallest set satisfying:*

- *each $p \in \mathcal{P}$ is an LTL formula; and*
- *if φ and ψ are LTL formulae, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\bigcirc\varphi$, $\Diamond\varphi$, $\Box\varphi$, $\varphi \,\mathcal{U}\, \psi$, and $\varphi \,\mathcal{R}\, \psi$.*

The closure of φ, denoted by $\text{cl}(\varphi)$*, is the set of its distinct subformulae, defined as:*

- $\mathrm{cl}(p) = \{p\}$, *if* $p \in \mathcal{P}$ *(atomic propositions);*
- $\mathrm{cl}(\mathrm{op}(\varphi)) = \{\mathrm{op}(\varphi)\} \cup \mathrm{cl}(\varphi)$, *if* $\mathrm{op} \in \{\neg, \bigcirc, \Diamond, \Box\}$ *(unary operators); and*
- $\mathrm{cl}(\mathrm{op}(\varphi, \psi)) = \{\mathrm{op}(\varphi, \psi)\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$, *if* $\mathrm{op} \in \{\wedge, \vee, \Rightarrow, \mathcal{U}, \mathcal{R}\}$ *(binary operators).*

The length of a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$*, defined as* $|\varphi| = \mathrm{card}(\mathrm{cl}(\varphi))$*, is the number of distinct subformulae it contains.*

For notational convenience, we have added syntactic sugar in the above definition by including the operators $\wedge$, $\Rightarrow$, $\Box$, $\Diamond$, $\mathcal{R}$ with their usual meaning as part of the syntax, although they can be defined using the $\neg$, $\vee$, and $\mathcal{U}$ operators.

Usually, the semantics of LTL are defined in terms of a satisfiability relation between an LTL formula over the set of atomic propositions $\mathcal{P}$ and an infinite word over $\Sigma = 2^{\mathcal{P}}$. Having in mind the rLTL version of these notions, we provide a mathematically equivalent definition of the semantics as a mapping W assigning an infinite word $\sigma \in \Sigma^{\omega}$ and an LTL formula φ to the element $W(\sigma, \varphi) \in \mathbb{B}$.

Definition 12 (LTL Semantics)**.** *The* LTL semantics *is a mapping* $W : \left(2^{\mathcal{P}}\right)^{\omega} \times \mathrm{LTL}(\mathcal{P}) \to \{0, 1\}$*, inductively defined as follows for* $p \in \mathcal{P}$ *and* $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$*:*

- *For atomic propositions:* $W(\sigma, p) = \begin{cases} 0, & \text{if } p \notin \sigma(0), \\ 1, & \text{if } p \in \sigma(0). \end{cases}$
- *For logical connectives:*

$$W(\sigma, \neg\varphi) = 1 - W(\sigma, \varphi), \qquad W(\sigma, \varphi \wedge \psi) = \min\{W(\sigma, \varphi), W(\sigma, \psi)\},$$
$$W(\sigma, \varphi \vee \psi) = \max\{W(\sigma, \varphi), W(\sigma, \psi)\}, \quad W(\sigma, \varphi \Rightarrow \psi) = \max\{W(\sigma, \neg\varphi), W(\sigma, \psi)\}.$$

- *For temporal operators:*

$$W(\sigma, \bigcirc\varphi) = W(\sigma_{1..}, \varphi), \quad W(\sigma, \Diamond\varphi) = \sup_{i\geq 0} W(\sigma_{i..}, \varphi), \quad W(\sigma, \Box\varphi) = \inf_{i\geq 0} W(\sigma_{i..}, \varphi),$$

$$W(\sigma, \varphi\, \mathcal{U}\, \psi) = \sup_{j\geq 0} \min \left\{ W(\sigma_{j..}, \psi), \inf_{0\leq i<j} W(\sigma_{i..}, \varphi) \right\},$$

$$W(\sigma, \varphi\, \mathcal{R}\, \psi) = \inf_{j\geq 0} \max \left\{ W(\sigma_{j..}, \psi), \sup_{0\leq i<j} W(\sigma_{i..}, \varphi) \right\}.$$

The evaluation of the mapping W on the LTL formula φ and the infinite word σ, $W(\sigma, \varphi)$, is the valuation of φ over σ.

Having introduced the semantics of LTL, we now recall the problem of *LTL model-checking* [CGP99, CHV18, KPR98, LP85, PZ08, Sch02, VW86], which is essential in formal in verification. Given a *model* of a system, the question is to decide whether or not all possible executions of the model satisfy a specification. Traditionally, these models are described by *Kripke structures* [CHV18, Section 2.2], and the specifications are described by LTL formulae.

Definition 13 (Kripke structures)**.** *A Kripke structure over a set $\mathcal{P}$ of atomic propositions is a quadruple $\mathcal{K} = (Q, Q_0, R, \ell)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $R \subseteq Q \times Q$ is a set of transitions, and $\ell : Q \to 2^{\mathcal{P}}$ is the labeling function that associates each state with a set of atomic propositions.*

A path *in $\mathcal{K}$ is an infinite sequence of states $q_0 q_1 \cdots \in Q^\omega$ such that $q_0 \in Q_0$ and $(q_i, q_{i+1}) \in R$ for all $i \in \mathbb{N}$. Any path induces a corresponding* computation $\ell(q_0)\ell(q_1)\cdots \in \left(2^{\mathcal{P}}\right)^\omega$.

In addition to Kripke structures, Büchi automata are another ingredient in the solution of the LTL model-checking problem. LTL formulae can be translated to *Büchi Automata* (BA) [BKL08, Section 4.3], [CHV18, Section 4.2] and the set of words they recognize.

Definition 14 (Büchi Automaton)**.** *A* (non-deterministic) Büchi Automaton *(BA) is a quintuple $\mathcal{A} = (Q, \Sigma, Q_0, \Delta, F)$ consisting of a nonempty, finite set Q of states, a (finite)*

input alphabet Σ, a set of initial states $Q_0 \subseteq Q$, a (nondeterministic) transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set $F \subseteq Q$ defining the acceptance conditions.

The run *of a BA on a word $\sigma \in \Sigma^\omega$ (also called* input*) is an infinite sequence of states $q_0 q_1 \ldots \in Q^\omega$ satisfying $q_0 \in Q_0$ and $(q_i, \sigma(i), q_{i+1}) \in \Delta$ for all $i \in \mathbb{N}$. A run ρ is called* accepting *if at least one of its infinitely often occurring states is in F. The* language *of a BA $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of all infinite words $\sigma \in \Sigma^\omega$ for which an accepting run of $\mathcal{A}$ exists.*

The translation of an LTL formula φ to a BA $\mathcal{A}_\varphi$ is done in two steps: 1) the LTL formula φ is translated to a *Generalized Büchi Automaton* (GBA); and 2) the GBA is translated to a BA.

Definition 15 (Generalized Büchi Automaton)**.** *A* (non-deterministic) Generalized Büchi Automaton *(GBA) is a quintuple $\mathcal{G} = (Q, \Sigma, Q_0, \Delta, \mathcal{F})$ consisting of a nonempty, finite set Q of states, a (finite) input alphabet Σ, a set of initial states $Q_0 \subseteq Q$, a (nondeterministic) transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set $\mathcal{F} \subseteq 2^Q$ denoting the acceptance conditions.*

The run *of a GBA is defined analogously to the run of a BA. The difference is that a run is* accepting *if its set of infinitely often occurring states contains at least one state from each accepting set in $\mathcal{F}$. Note that there may be no accepting sets, in which case any infinite run trivially satisfies this property.*

Transforming a GBA $\mathcal{G}$ into an equivalent BA $\mathcal{A}$ requires creating $\mathrm{card}(\mathcal{F})$ many copies of $\mathcal{G}$. The acceptance set of copy $\mathcal{G}_i$ is connected to the states of copy $\mathcal{G}_{\mathrm{mod}(i+1,\mathrm{card}(\mathcal{F}))}$, $i = 1, \ldots, \mathrm{card}(\mathcal{F})$. Then, the accepting condition for $\mathcal{A}$ asks that any accepting state of the first copy is visited infinitely often. This implies that the accepting sets of each copy are visited infinitely often too. For more details see [BKL08, Section 4.3.4].

Proposition 10.0.1 (Section 5.2, [BKL08])**.** *Any LTL formula φ can be translated to a GBA $\mathcal{G}_\varphi$ with at most $2^{|\varphi|}$ states and at most $|\varphi|$ accepting conditions. Moreover, the GBA $\mathcal{G}_\varphi$ can be translated to a BA $\mathcal{A}_\varphi$ with at most $|\varphi| \cdot 2^{|\varphi|}$ states.*

Therefore, the time complexity of translating an LTL formula φ to a GBA $\mathcal{A}_\varphi$ is $\mathcal{O}\left(2^{|\varphi|}\right)$, and that of translating an LTL formula φ to a BA $\mathcal{A}_\varphi$ is $\mathcal{O}\left(|\varphi| \cdot 2^{|\varphi|}\right)$.

Problem 4 (LTL model-checking)**.** *Given a set of atomic propositions $\mathcal{P}$, a Kripke structure $\mathcal{K}$ and an LTL formula φ, do all the computations of $\mathcal{K}$ satisfy φ?*

The classical approach to solving Problem 4 has running time depending linearly on the size of the Kripke structure and exponentially on the length of the LTL formula.

Corollary 10.0.2 (LTL model-checking)**.** *The standard procedure for model-checking an LTL formula φ on a Kripke structure $\mathcal{K}$ is as follows [BKL08, Section 5.2], [CHV18, Section 4]:*

1. *Construct a BA $\mathcal{A}_\mathcal{K}$ such that $\mathcal{A}_\mathcal{K}$ accepts a computation $\pi \in \left(2^\mathcal{P}\right)^\omega$ if and only if π is a computation of $\mathcal{K}$. The size of $\mathcal{A}_\mathcal{K}$ is linear in the size of $\mathcal{K}$, i.e., in its number of states denoted by $|\mathcal{K}|$.*

2. *Construct a BA $\mathcal{A}_{\neg\varphi}$ recognizing the words satisfying the negation of φ, i.e., $\neg\varphi$. The size of $\mathcal{A}_{\neg\varphi}$ is exponential in $|\varphi|$, specifically $\mathcal{O}\left(|\varphi| \cdot 2^{|\varphi|}\right)$ by Proposition 10.0.1.*

3. *Compose $\mathcal{A}_\mathcal{K}$ with $\mathcal{A}_{\neg\varphi}$ to obtain $\mathcal{A}_{\mathcal{K},\neg\varphi}$, which recognizes* all the words of $L\left(\mathcal{A}_\mathcal{K}\right)$ that do not satisfy φ, *i.e., $L\left(\mathcal{A}_{\mathcal{K},\neg\varphi}\right) = L\left(\mathcal{A}_\mathcal{K}\right) \cap L\left(\mathcal{A}_{\neg\varphi}\right)$. The size of $\mathcal{A}_{\mathcal{K},\neg\varphi}$ is $\mathcal{O}\left(|\mathcal{K}| \cdot |\varphi| \cdot 2^{|\varphi|}\right)$.*

4. *Check the* emptiness *of $L\left(\mathcal{A}_{\mathcal{K},\neg\varphi}\right)$: if $\mathcal{A}_{\mathcal{K},\neg\varphi}$ only recognizes the empty language, then $L(\mathcal{A}_\mathcal{K})$ satisfies φ.*

The time complexity of Step 4 in terms of the size of $\mathcal{K}$ and the length of the LTL formula φ is:

$$\mathcal{O}\Big(|\mathcal{K}| \cdot |\varphi| \cdot 2^{|\varphi|}\Big). \tag{10.1}$$

The above represents the classical, and tight, upper bound for the time complexity of LTL model-checking.

In this work, we are concerned with solutions to the model-checking problem that employ a translation of formulae to GBAs as we described in the previous paragraph. For this reason, when discussing the complexity of the model-checking problem induced by different temporal logics, we focus on the size of the corresponding GBA. For example, for an LTL formula φ the corresponding GBA has size $\mathcal{O}\left(2^{|\varphi|}\right)$, and in Sections 12 and 13 we will derive similar upper bounds for rLTL and an rLTL fragment respectively.

CHAPTER 11

The Syntax and Semantics of Robust Linear Temporal Logic

The main goal of *robust Linear Temporal Logic* (rLTL) is to embed a notion of robustness into LTL. With this in mind, we crafted the syntax of rLTL so as to closely resemble that of LTL by using *robust* versions of LTL operators. In addition to defining the rLTL syntax, we also define the rLTL semantics in this section and justify the necessity of the many-valued semantics, i.e., the five rLTL truth values. We do so by first considering the $\text{rLTL}_{\boxdot,\diamondsuit}(\mathcal{P})$ fragment, i.e., the fragment of rLTL formulae that only allows the "robust always" $\boxdot$ and "robust eventually" $\diamondsuit$ temporal operators. This fragment was first introduced in [TN16], and in this section we extend those results from the fragment $\text{rLTL}_{\boxdot,\diamondsuit}(\mathcal{P})$ to full rLTL.

11.1 rLTL Syntax

We begin by presenting the rLTL syntax.

Definition 16 (rLTL syntax)**.** *Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. The set of all* rLTL formulae *on $\mathcal{P}$, written* $\text{rLTL}(\mathcal{P})$*, is the smallest set satisfying:*

- *each $p \in \mathcal{P}$ is an rLTL formula; and*
- *if φ and ψ are rLTL formulae, then so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rrightarrow \psi$, $\odot\varphi$, $\diamondsuit\varphi$, $\boxdot\varphi$, $\varphi \mathcal{U} \psi$, and $\varphi \mathcal{R} \psi$.*

The length of a $\varphi \in \text{rLTL}(\mathcal{P})$, denoted by $|\varphi|$, is the number of its distinct subformulae.

Notice that in LTL, the conjunction and implication operators can be derived from negation and disjunction. This is no longer the case in rLTL since it has a many-valued semantics. On these grounds, we directly included conjunction and robust implication as part of the rLTL syntax in Definition 16. The same reason justifies the presence of the robust release operator $\mathcal{R}$, which, in the case of LTL, can be derived from the until and negation operators as $\varphi \,\mathcal{R}\, \psi = \neg\,(\neg\varphi \,\mathcal{U}\, \psi)$.

11.2 The $\mathrm{rLTL}_{\boxdot,\diamondsuit}(\mathcal{P})$ fragment

Consider, as a running example, the LTL formula $\Box p$ with p an atomic proposition. There is *only one way* in which this formula can be satisfied, namely when p holds at every time step. In contrast, there are *several ways* in which this formula can be violated. Our goal is to find a semantics that distinguishes between these different ways. We aim for such distinction to be limited by what can be expressed in LTL so that we can easily leverage the wealth of existing results on LTL verification and synthesis.

By intuitively investigating the different ways in which $\Box p$ is violated over an infinite trace, we are able to distinguish the following four cases: (1) the worst possible violation occurs when p fails to hold at every time step; (2) a slightly better scenario, which still violates $\Box p$, is where p holds for at most finitely many time instants; (3) better yet would be that p holds at infinitely many instants, while still failing to hold at infinitely many instants; (4) finally, among all the possible ways in which $\Box p$ can be violated, the most preferable case would be the one where p fails to hold for at most finitely many time instants. Consequently, our robust semantics is designed to distinguish between satisfaction and these four possible different ways to violate $\Box p$. However, as convincing as this argument might be, a question persists: in which sense can we regard these five alternatives as canonical?

11.2.1 Why 5 truth values?

We answer this question by interpreting satisfaction of $\Box p$ as a counting problem, and showing that there exists a partition of the infinite strings $\mathbb{B}^\omega = \{0,1\}^\omega$, which is induced by the $\mathrm{LTL}_{\Box,\Diamond}$ formulae. The semantics of the LTL $\Box$ operator is given by:

$$W(\sigma, \Box\varphi) = \inf_{i \in \mathbb{N}} W(\sigma_{i..}, \varphi). \tag{11.1}$$

We make the following observation: the truth value of the LTL formula $\Box\varphi$ on the infinite word $\sigma \in \Sigma^\omega$ is invariant under permutations of $W^\omega(\sigma, \varphi) = W(\sigma_{0..}, \varphi)W(\sigma_{1..}, \varphi)\cdots$. To make this observation precise, let $f : \mathbb{N} \to \mathbb{N}$ be a permutation of $\mathbb{N}$, i.e., a bijection. Then, we have:

$$\inf_{i \in \mathbb{N}} W(\sigma_{i..}, \varphi) = \inf_{i \in \mathbb{N}} W(\sigma_{f(i)..}, \varphi). \tag{11.2}$$

The above property shows that the $\Box$ operator counts the number of zeros in the infinite string $W^\omega(\sigma, \varphi) \in \{0,1\}^\omega$. When this number is 0, $W(\sigma, \Box\varphi)$ is one (*true*), and otherwise zero (*false*). The position where zeros occur is not relevant, only their presence or their absence is. Towards characterizing how to count in $\mathrm{LTL}_{\Box,\Diamond}$, we first note that by successively applying permutations that swap position i with position $i+1$ and leave all the remaining elements of $\mathbb{N}$ unaltered, we can transform any string $\rho \in \{0,1\}^\omega$ into one of the following forms $1^\omega,\ 0^k1^\omega,\ (01)^\omega,\ 1^k0^\omega,\ 0^\omega$, where $k \in \mathbb{N}$. Moreover, since $\mathrm{LTL}_{\Box,\Diamond}$ is stutter-free [PW97], it follows that the previous cases degenerate into the following five:

$$1^\omega,\ 01^\omega,\ (01)^\omega,\ 10^\omega,\ \text{and}\ 0^\omega. \tag{11.3}$$

We interpret these as the ability to count if the number of zeros and ones is *zero*, *finite*, or *infinite*. If we denote by (z, o) the number of zeros and ones, with $z, o \in \{\mathrm{zer}, \mathrm{fin}, \mathrm{inf}\}$, then we have:

- 1^ω corresponds to $(\mathrm{zer}, \mathrm{inf})$.

- 01^ω corresponds to (fin, inf).
- $(01)^\omega$ corresponds to (inf, inf).
- 10^ω corresponds to (inf, fin).
- 0^ω corresponds to (inf, zer).

We can thus conclude the need for 5 truth values to describe the 5 different ways of counting zeros and ones. Concretizing this discussion, in particular (11.3), on the running example of the formula $\Box p$, we obtain the following canonical forms that can be distinguished:

$$\{p\}^\omega, \quad (\{\neg p\}\{p\})^+ \{p\}^\omega, \quad (\{\neg p\}\{p\})^\omega, \quad (\{\neg p\}\{p\})^+ \{\neg p\}^\omega, \quad \text{and } \{\neg p\}^\omega. \tag{11.4}$$

It is no surprise that these are exactly the five cases discussed in the beginning of this subsection.

The considerations in this section suggest the need for a semantics that is 5-valued rather than 2-valued so that we can distinguish between the aforementioned five cases. Therefore, we need to replace Boolean algebras by a different type of algebraic structure that can accommodate a 5-valued semantics. *Da Costa algebras*, reviewed in the next section, are an example of such algebraic structures.

11.2.2 da Costa Algebras

According to our running example $\Box p$, the desired semantics should have one truth value corresponding to *true* and four truth values corresponding to different shades of *false*. It is instructive to think of truth values as the elements of $\mathbb{B}^4$, i.e., the four-fold Cartesian product of $\mathbb{B}$, that arise as the possible values of the 4-tuple of LTL formulae:

$$(\Box p, \Diamond\Box p, \Box\Diamond p, \Diamond p). \tag{11.5}$$

To ease notation, we denote such values interchangeably by $b = b_1b_2b_3b_4$ and $b = (b_1, b_2, b_3, b_4)$ with $b_i \in \mathbb{B}$ for $i \in \{1, 2, 3, 4\}$. The value 1111 then corresponds to *true*

since $\Box p$ is satisfied. The most preferred violation of $\Box p$ (p fails to hold at only finitely many time instants) corresponds to 0111, followed by 0011 (p holds at infinitely many instants and also fails to hold at infinitely many instants), 0001 (p holds at most at finitely many instants), and 0000 (p fails to hold at every time instant). Such preferences can be encoded in the linear order:

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111, \tag{11.6}$$

that renders the set:

$$\mathbb{B}_5 = \{0000, 0001, 0011, 0111, 1111\}, \tag{11.7}$$

a (bounded) distributive lattice with top element $\top = 1111$ and bottom element $\bot = 0000$. Formally, $\mathbb{B}_5$ is the subset of $\mathbb{B}^4$ consisting of the 4-tuples $(b_1, b_2, b_3, b_4) \in \mathbb{B}^4$ satisfying the monotonicity property:

$$i \leq_{\mathbb{N}} j \text{ implies } b_i \leq_{\mathbb{B}} b_j, \tag{11.8}$$

where $i, j \in \{1, \ldots, 4\}$, $\leq_{\mathbb{N}}$ is the natural order on the natural numbers and $\leq_{\mathbb{B}}$ is the natural order on the Boolean algebra $\mathbb{B}$. In $\mathbb{B}_5$, the meet $\sqcap$ can be interpreted as minimum and the join $\sqcup$ as maximum with respect to the order in (11.6). We use $\sqcap$ and $\sqcup$ when discussing lattices in general and use min and max for the specific lattice $\mathbb{B}_5$ or the Boolean algebra $\mathbb{B}$.

The first choice to be made in using the lattice $(\mathbb{B}_5, \min, \max)$ to define the semantics of $\mathrm{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ is the choice of an operation on $\mathbb{B}_5$ modeling conjunction. It is well know that all the desirable properties of a many-valued conjunction are summarized by the notion of triangular-norm, see [H98, NPM99]. One can compare two triangular-norms s and t using the partial order defined by declaring $s \leq t$ when $s(a, b) \leq t(a, b)$ for all $a, b \in \mathbb{B}_5$. According to this order, the triangular-norm min is maximal among all triangular-norms (i.e., we have $t(a, b) \leq \min\{a, b\}$ for every $a, b \in \mathbb{B}_5$ and every triangular-norm t). This shows that if we choose any triangular-norm t different from min, there exist elements $a, b \in \mathbb{B}_5$ for which we have $t(a, b) < \min\{a, b\}$. Hence, any choice different from min would result in situations

where the value of a conjunction is *smaller* than the value of the conjuncts, which is not reasonable when interpreting the value of the conjuncts as different shades of *false*. To illustrate this point, consider the formula $\Box p \wedge \Box q$ and the word $\sigma = \emptyset\{p, q\}^\omega$. As introduced above, the value of $\Box p$ on σ corresponds to 0111 and the value of $\Box q$ on σ corresponds to 0111 since on both cases we have the most preferred violation of the formulae. Therefore, the value of $\Box p \wedge \Box q$ on σ should also be 0111 since the formula $\Box p \wedge \Box q$ is only violated a finite number of times. It thus seems natural[1] to model conjunction in $\mathbb{B}_5$ by min and, for similar reasons, to model disjunction in $\mathbb{B}_5$ by max.

As in intuitionistic logic, our implication is defined as the residue of $\sqcap$[2]. In other words, we define the implication $a \to b$ by requiring that $c \preceq a \to b$ if and only if $c \sqcap a \preceq b$ for every $c \in \mathbb{B}_5$. This leads to:

$$a \to b = \begin{cases} 1111 & \text{if } a \preceq b; \text{ and} \\ b & \text{otherwise.} \end{cases}$$

However, we now *diverge* from intuitionistic logic (and most many-valued logics) where negation of a is defined by $a \to 0000$. Such negation is not compatible with the interpretation that all the elements of $\mathbb{B}_5$, except for 1111, represent (different shades of) *false* and thus their negation should have the truth value 1111. To make this point clear, we present in Table 11.1 the intuitionistic negation in $\mathbb{B}_5$ and the desired negation compatible with the interpretation of the truth values in $\mathbb{B}_5$.

---

[1]Note that there are situations where it is convenient to model conjunction differently. In the work of Bloem et al. [BCG10], the specific way in which robustness is modeled requires distinguishing between the number of conjuncts that are satisfied in the assumption $\wedge_{i \in I} \varphi_i$. This cannot be accomplished if conjunction is modeled by min, and a different triangular-norm would have to be used for this purpose. Note that both Łukasiewicz's conjunction as well as Goguen's conjunction have the property that their value decreases as the number of conjuncts that are true decreases.

[2]This is also done in context of residuated lattices that is more general than the Heyting algebras used in intuitionistic logic. Recall that a residuated lattice is a lattice $(A, \sqcap, \sqcup)$, satisfying the same additional conditions, and equipped with a commutative monoid $(A, \otimes, \mathbf{1})$ satisfying additional compatibility conditions. Since we chose the lattice meet $\sqcap$ to represent conjunction, we have a residuated lattice where $\otimes = \sqcap$ and $\mathbf{1} = \top$.

Table 11.1: Desired negation vs. intuitionistic negation in $\mathbb{B}_5$.

Value	Desired negation	Intuitionistic negation
1111	0000	0000
0111	1111	0000
0011	1111	0000
0001	1111	0000
0000	1111	1111

What is then the algebraic structure on $\mathbb{B}_5$ that supports the desired negation, dual to the intuitionistic negation? This very same problem was investigated in [Pri09], and the answer is *da Costa* algebras.

Definition 17 (da Costa algebra)**.** *A* da Costa *algebra is a sextuple* $(A, \sqcap, \sqcup, \preceq, \to, \overline{\cdot})$ *where:*

- $(A, \sqcap, \sqcup, \preceq)$ *is a distributive lattice where* $\preceq$ *is the ordering relation derived from* $\sqcap$ *and* $\sqcup$*,*
- $\to$ *is the residual of* $\sqcap$ *(i.e.,* $a \preceq b \to c$ *if and only if* $a \sqcap b \preceq c$ *for every* $a, b, c \in A$*),*
- $a \preceq b \sqcup \overline{b}$ *for every* $a, b \in A$*, and*
- $\overline{a} \preceq b$ *whenever* $c \sqcup \overline{c} \preceq a \sqcup b$ *for every* $a, b, c \in A$*.*

In a da Costa algebra, one can define the top element $\top$ to be $\top = a \sqcup \overline{a}$ for an arbitrary $a \in A$; note that $\top$ is unique and independent of the choice of a. Hence, the third requirement in Definition 17 amounts to the definition of top element, while the fourth requirement can be simplified to $\overline{a} \preceq b$ whenever $\top \preceq a \sqcup b$. We can easily verify that $\mathbb{B}_5$ is a da Costa algebra if we use the desired negation defined in Table 11.1.

It should be mentioned that working with a 5-valued semantics has its price. The law of non-contradiction fails in $\mathbb{B}_5$ (i.e., $a \sqcap \overline{a}$ may not equal $\bot = 0000$ as evidenced by taking

$a = 0111$). However, since $a \sqcap \overline{a} \prec 1111$, a weak form of non-contradiction still holds as $a \sqcap \overline{a}$ is to be interpreted as a shade of *false* but not necessarily as the least preferred way of violating $a \sqcap \overline{a}$, which corresponds to $\bot$. Contrary to intuitionistic logic, the law of excluded middle is valid (i.e., $a \sqcup \overline{a} = \top = 1111$). Finally, $a = 0111$ shows that $\overline{\overline{a}} \neq a$, although it is still true that $\overline{\overline{a}} \rightarrow a$. Interestingly, we can think of the double negation:

$$\overline{\overline{a}} = \begin{cases} 1111, & \text{if } a = 1111, \\ 0000, & \text{otherwise,} \end{cases}$$

as quantization in the sense that *true* is mapped to *true* and all the shades of *false* are mapped to *false*. Hence, double negation quantizes the five different truth values into two truth values (*true* and *false*) in a manner that is compatible with our interpretation of truth values.

11.2.3 Semantics of $\mathrm{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ on da Costa Algebras

The semantics of $\mathrm{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ is given by a mapping V, called valuation as in the case of LTL, that maps an infinite word $\sigma \in \Sigma^\omega$ and an $\mathrm{rLTL}_{\boxdot,\diamond}(\mathcal{P})$ formula φ to an element of $\mathbb{B}_5$. In defining V, we judiciously use the algebraic operations of the da Costa algebra $\mathbb{B}_5$ to give meaning to the logical connectives in the syntax of $\mathrm{rLTL}_{\boxdot,\diamond}(\mathcal{P})$. In the following, let $\Sigma = 2^{\mathcal{P}}$, where $\mathcal{P}$ is a finite set of atomic propositions.

The semantics of rLTL is a mapping $V : \left(2^{\mathcal{P}}\right)^\omega \times \mathrm{rLTL}(\mathcal{P}) \rightarrow \mathbb{B}_5$. On atomic propositions $p \in \mathcal{P}$, V is defined by:

$$V(\sigma, p) = \begin{cases} 0000, & \text{if } p \notin \sigma(0); \text{ and} \\ 1111, & \text{if } p \in \sigma(0). \end{cases} \tag{11.9}$$

Hence, atomic propositions are interpreted classically, i.e., only two truth values are used. Since we are using a 5-valued semantics, we provide a separate definition for all the four

logical connectives:

$$V(\sigma, \neg\varphi) = \overline{V(\sigma, \varphi)}, \tag{11.10}$$

$$V(\sigma, \varphi \wedge \psi) = V(\sigma, \varphi) \sqcap V(\sigma, \psi), \tag{11.11}$$

$$V(\sigma, \varphi \vee \psi) = V(\sigma, \varphi) \sqcup V(\sigma, \psi), \tag{11.12}$$

$$V(\sigma, \varphi \Rrightarrow \psi) = V(\sigma, \varphi) \to V(\sigma, \psi). \tag{11.13}$$

Note how the semantics mirrors the algebraic structure of da Costa algebras. This is no accident since valuations are typically algebra homomorphisms.

Unfortunately, da Costa algebras are not equipped[3] with operations corresponding to $\boxdot$ and $\diamondsuit$, the robust versions of $\Box$ and $\Diamond$, respectively. Therefore, we resort to the counting interpretation in Section 11.2.1 to motivate the semantics of $\boxdot$. Formally, the semantics of $\boxdot$ is given by:

$$V(\sigma, \boxdot\varphi) = \left(\inf_{i \geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{j \geq 0} \inf_{i \geq j} V_2(\sigma_{i..}, \varphi),\ \inf_{j \geq 0} \sup_{i \geq j} V_3(\sigma_{i..}, \varphi),\ \sup_{i \geq 0} V_4(\sigma_{i..}, \varphi) \right), \tag{11.14}$$

where $V_k(\sigma, \varphi) = \pi_k \circ V(\sigma, \varphi)$ for $k \in \{1, 2, 3, 4\}$ and $\pi_k : \mathbb{B}_5 \to \mathbb{B}$ are the projections on the k-th element of a truth value defined by:

$$\pi_k(a_1, a_2, a_3, a_4) = a_k. \tag{11.15}$$

To illustrate the semantics of $\boxdot$, let us consider the simple case where φ is just an atomic proposition p. This means that one can express $V(\sigma, \boxdot p)$ in terms of the LTL valuation W by:

$$V(\sigma, \boxdot p) = \left(W(\sigma, \Box p), W(\sigma, \Diamond\Box p), W(\sigma, \Box\Diamond p), W(\sigma, \Diamond p)\right). \tag{11.16}$$

In other words, $V_1(\sigma, \boxdot p)$ corresponds to the LTL truth value of $\Box p$, $V_2(\sigma, \boxdot p)$ corresponds to the LTL truth value of $\Diamond\Box p$, $V_3(\sigma, \boxdot p)$ corresponds to the LTL truth value of $\Box\Diamond p$, and

[3]One could consider developing a notion of da Costa algebras with operators in the spirit of Boolean algebras with operators [JT51]. We leave such investigation for future work.

$V_4(\sigma, \boxdot p)$ corresponds to the LTL truth value of $\Diamond p$. Equation (11.16) connects the semantics of $\boxdot$ to the counting problems described in Section 11.2.1 and to the 4-tuple of LTL formulae in (11.5).

The last operator is $\diamondsuit$, whose semantics is given by:

$$V(\sigma, \diamondsuit\varphi) = \left(\sup_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_2(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi) \right). \tag{11.17}$$

According to the counting problems used in Section 11.2.1 to motivate the proposed semantics, there is only one way in which the LTL formula $\Diamond p$, for an atomic proposition p, can be violated. Hence, $V(\sigma, \diamondsuit p)$ is one of only two possible truth values: 1111 or 0000. We further note that $\diamondsuit$ is not dual to $\boxdot$, as expected in a many-valued logic where the law of double negation fails.

Having defined the semantics of $\text{rLTL}_{\boxdot,\diamondsuit}(\mathcal{P})$, let us now see if the formula $\boxdot p \Rrightarrow \boxdot q$, where $\boxdot p$ is an environment assumption and $\boxdot q$ is a system guarantee with $p, q \in \mathcal{P}$, lives to the expectations set in the introduction and to the intuition provided in Section 11.2.1.

1. According to (11.16), if $\Box p$ holds, then $\boxdot p$ evaluates to 1111 and the implication $\boxdot p \Rrightarrow \boxdot q$ is *true*, i.e., the value of $\boxdot p \Rrightarrow \boxdot q$ is 1111, if $\boxdot q$ evaluates to 1111, that is, if $\Box q$ holds. Therefore, the desired behavior of $\Box p \Rightarrow \Box q$, when the environment assumptions hold, is retained.

2. Consider now the case where $\Box p$ fails but the weaker assumption $\Diamond\Box p$ holds. In this case $\boxdot p$ evaluates to 0111 and the implication $\boxdot p \Rrightarrow \boxdot q$ is *true* if $\boxdot p$ evaluates to 0111 or higher. This means that $\Diamond\Box q$ needs to hold.

3. A similar argument shows that we can also conclude the following consequences whenever $\boxdot p \Rrightarrow \boxdot q$ evaluates to 1111: $\Box\Diamond q$ follows whenever the environment satisfies $\Box\Diamond p$ and $\Diamond q$ follows whenever the environment satisfies $\Diamond p$. We thus conclude that the semantics of $\boxdot p \Rrightarrow \boxdot q$ captures the desired robustness property by which a weakening of the assumption $\boxdot p$ leads to a weakening of the guarantee $\boxdot q$.

11.3 Full rLTL($\mathcal{P}$)

In this section, we present the semantics of full rLTL($\mathcal{P}$) by providing the semantics for three additional operators: the "robust release", denoted by $\mathcal{R}$, the "robust until", denoted by $\mathcal{U}$, and the "next", denoted by $\odot$, operators.

11.3.1 Why 5 truth values?

We revisit this question to better motivate the full rLTL semantics. According to the safety-progress classification of temporal properties, eloquently put forward in [CMP93, MP87], $\Box p$ defines a safety property. It can be expressed as $A(L)$ with L being the regular language $\Sigma^*\{p\}$ and A the operator generating all the infinite words in $\left(2^{\mathcal{P}}\right)^{\omega}$ with the property that all its finite prefixes belong to L. In addition to A, we can find in [CMP93, MP87] the operators E, R, and P defining guarantee, response, and persistence properties, respectively. The language $E(L)$ consists of all the infinite words that contain at least one prefix in L, the language $R(L)$ consists of all the infinite words that contain infinitely many prefixes in L, and the language $P(L)$ consists of all the infinite words such that all but finitely many prefixes belong to L. Using these operators we can reformulate the semantics of $\boxdot p$ from (11.14) as:

$$V(\sigma, \boxdot p) = \begin{cases} 1111 & \text{if } \sigma \in A(L); \\ 0111 & \text{if } \sigma \in P(L) \setminus A(L); \\ 0011 & \text{if } \sigma \in R(L) \setminus (A(L) \cup P(L)); \\ 0001 & \text{if } \sigma \in E(L) \setminus (A(L) \cup P(L) \cup R(L)); \text{ and} \\ 0000 & \text{if } \sigma \notin E(L). \end{cases} \tag{11.18}$$

We thus obtain a different justification for the five different truth values used in rLTL and why the five different cases in (11.4) can be seen as canonical. Moreover, we can build on this perspective to define the semantics of the robust release and the robust until.

11.3.2 Semantics of full rLTL($\mathcal{P}$)

We are now ready to define the semantics for the "robust release" $\mathcal{R}$, and "robust until" $\mathcal{U}$ operators. Equality (11.18) suggests how we can define the 5-valued semantics for the release operator. Recall that the LTL formula $p\,\mathcal{R}\,q$, for atomic propositions p and q, defines a safety property, and that its semantics is given by:

$$W(\sigma, p\ \mathcal{R}\ q) = \inf_{j\geq 0} \max\left\{V_1(\sigma_{j..}, q),\ \sup_{0\leq i<j} V_1(\sigma_{i..}, p)\right\}. \tag{11.19}$$

We can interpret $\max\left\{V_1(\sigma_{j..}, q), \sup_{0\leq i<j} V_1(\sigma_{i..}, p)\right\}$ above as the definition of the regular language $L = \Sigma^*\{q\} + \Sigma^*\{p\}\Sigma^+$ and $\inf_{j\geq 0}$ as the requirement that every prefix of a string satisfying $p\,\mathcal{R}\,q$ belongs to L, i.e., as the definition of the operator A. Therefore, the 5-valued semantics can be obtained by successively enlarging the language $A(L)$ through the replacement of the operator A, formalized by inf in Equation (11.19), by the operators P formalized by sup inf, R formalized by inf sup, and E for formalized by sup. This observation leads to the semantics:

$$V(\sigma, \varphi\ \mathcal{R}\ \psi) = (V_1(\sigma, \varphi\ \mathcal{R}\ \psi), V_2(\sigma, \varphi\ \mathcal{R}\ \psi), V_3(\sigma, \varphi\ \mathcal{R}\ \psi), V_4(\sigma, \varphi\ \mathcal{R}\ \psi)), \tag{11.20}$$

where:

$$\begin{aligned}
V_1(\sigma, \varphi\ \mathcal{R}\ \psi) &= \inf_{j\geq 0} \max\left\{V_1(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_1(\sigma_{i..}, \varphi)\right\},\\
V_2(\sigma, \varphi\ \mathcal{R}\ \psi) &= \sup_{k\geq 0} \inf_{j\geq k} \max\left\{V_2(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_2(\sigma_{i..}, \varphi)\right\},\\
V_3(\sigma, \varphi\ \mathcal{R}\ \psi) &= \inf_{k\geq 0} \sup_{j\geq k} \max\left\{V_3(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_3(\sigma_{i..}, \varphi)\right\},\\
V_4(\sigma, \varphi\ \mathcal{R}\ \psi) &= \sup_{j\geq 0} \max\left\{V_4(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_4(\sigma_{i..}, \varphi)\right\}.
\end{aligned}$$

Similarly to LTL, $\boxdot\psi = \mathit{false}\ \mathcal{R}\ \psi$ holds, thereby showing that the semantics for the $\mathcal{R}$ operator is compatible with the semantics of the $\boxdot$ operator. We glean further intuition behind the definition of $\mathcal{R}$ by considering the special case where φ is given by p and ψ is

given by q, for two atomic propositions $p, q \in \mathcal{P}$. Expressing $V(\sigma, p\,\mathcal{R}\,q)$ in terms of an LTL valuation W, we obtain:

$$V(\sigma, p\,\mathcal{R}\,q) = (W(\sigma, p\,\mathcal{R}\,q),\ W(\sigma, \Diamond\Box q \vee \Diamond p),\ W(\sigma, \Box\Diamond q \vee \Diamond p),\ W(\sigma, \Diamond q \vee \Diamond p))\,.$$

As long as p occurs, the value of $p\ \mathcal{R}\ q$ is at least 0111. It could be argued that the semantics of $p\ \mathcal{R}\ q$ should also count the number of occurrences of q preceding the first occurrence of p. As we detail in Section 11.5, this can be expressed in rLTL by making use of the proposed semantics.

In LTL, the $\mathcal{U}$ operator is dual to the $\mathcal{R}$ operator, but such relationship does not extend to rLTL in virtue of how negation was defined. Hence, the semantics of the $\mathcal{U}$ operator has to be introduced separately. Analogously to above, we interpret the LTL semantics of $p\,\mathcal{U}\,q$, given by:

$$W(\sigma, p\,\mathcal{U}\,q) = \sup_{j\geq 0} \min\left\{V_1(\sigma_{j..}, q), \inf_{0\leq i<j} V_1(\sigma_{i..}, p)\right\}, \tag{11.21}$$

as defining the language $E\left(\{p\}^*\{q\}\right)$. In the hierarchy of the operators E, R, P, and A, defined by the inclusions $A(L) \subset P(L) \subset R(L) \subset E(L)$ for any regular language L, the language $E\left(\{p\}^*\{q\}\right)$ cannot be enlarged as it sits at the top of the hierarchy. Therefore, the semantics of the $\mathcal{U}$ operator is given by:

$$V(\sigma, \varphi\,\mathcal{U}\,\psi) = (V_1(\sigma, \varphi\,\mathcal{U}\,\psi), V_2(\sigma, \varphi\,\mathcal{U}\,\psi), V_3(\sigma, \varphi\,\mathcal{U}\,\psi), V_4(\sigma, \varphi\,\mathcal{U}\,\psi))\,, \tag{11.22}$$

where:

$$V_k(\sigma, \varphi\,\mathcal{U}\,\psi) = \sup_{j\geq 0} \min\left\{V_k(\sigma_{j..}, \psi), \inf_{0\leq i<j} V_k(\sigma_{i..}, \varphi)\right\},\ \text{for each } k \in \{1,2,3,4\}.$$

We obtain, by definition, that the semantics of the $\mathcal{U}$ operator is compatible with the semantics of the $\diamond$ operator in the sense that $\diamond\psi = \mathit{true}\,\mathcal{U}\,\psi$.

Finally, we define the robust semantics of next as a direct generalization of the LTL semantics from $\mathbb{B}$ to $\mathbb{B}_5$:

$$V(\sigma, \odot\varphi) = V(\sigma_{1..}, \varphi). \tag{11.23}$$

11.4 Review of rLTL semantics

To recap the previous discussion, we compactly present the formal rLTL semantics below.

Definition 18 (rLTL Semantics)**.** *The* rLTL semantics *is a mapping* $V : \left(2^{\mathcal{P}}\right)^{\omega} \times \mathrm{rLTL}(\mathcal{P}) \to \mathbb{B}_4$*, inductively defined as follows for* $p \in \mathcal{P}$ *and* $\varphi, \psi \in \mathrm{rLTL}(\mathcal{P})$*:*

- *For atomic propositions:* $V(\sigma, p) = \begin{cases} 0000, & \text{if } p \notin \sigma(0), \\ 1111, & \text{if } p \in \sigma(0). \end{cases}$
- *For logical connectives:*

$$V(\sigma, \neg\varphi) = \overline{V(\sigma, \varphi)}, \qquad V(\sigma, \varphi \wedge \psi) = V(\sigma, \varphi) \sqcap V(\sigma, \psi),$$
$$V(\sigma, \varphi \vee \psi) = V(\sigma, \varphi) \sqcup V(\sigma, \psi), \qquad V(\sigma, \varphi \Rrightarrow \psi) = V(\sigma, \varphi) \to V(\sigma, \psi).$$

- *For temporal operators:*

$$V(\sigma, \odot\varphi) = V(\sigma_{1..}, \varphi),$$
$$V(\sigma, \diamondsuit\varphi) = \left(\sup_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_2(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\right),$$
$$V(\sigma, \boxdot\varphi) = \left(\inf_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{j\geq 0}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi),\ \inf_{j\geq 0}\sup_{i\geq j} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\right),$$
$$V(\sigma, \varphi\, \mathcal{U}\, \psi) = \left(V_1(\sigma, \varphi\, \mathcal{U}\, \psi), V_2(\sigma, \varphi\, \mathcal{U}\, \psi), V_3(\sigma, \varphi\, \mathcal{U}\, \psi), V_4(\sigma, \varphi\, \mathcal{U}\, \psi)\right),$$

where: $V_k(\sigma, \varphi\, \mathcal{U}\, \psi) = \sup_{j\geq 0} \min\left\{V_k(\sigma_{j..}, \psi), \inf_{0\leq i<j} V_k(\sigma_{i..}, \varphi)\right\}, k \in \{1, 2, 3, 4\}.$

$$V(\sigma, \varphi\, \mathcal{R}\, \psi) = \left(V_1(\sigma, \varphi\, \mathcal{R}\, \psi), V_2(\sigma, \varphi\, \mathcal{R}\, \psi), V_3(\sigma, \varphi\, \mathcal{R}\, \psi), V_4(\sigma, \varphi\, \mathcal{R}\, \psi)\right),$$

where:

$$V_1(\sigma, \varphi\, \mathcal{R}\, \psi) = \inf_{j\geq 0} \max\left\{V_1(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_1(\sigma_{i..}, \varphi)\right\},$$
$$V_2(\sigma, \varphi\, \mathcal{R}\, \psi) = \sup_{k\geq 0}\inf_{j\geq k} \max\left\{V_2(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_2(\sigma_{i..}, \varphi)\right\},$$
$$V_3(\sigma, \varphi\, \mathcal{R}\, \psi) = \inf_{k\geq 0}\sup_{j\geq k} \max\left\{V_3(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_3(\sigma_{i..}, \varphi)\right\},$$
$$V_4(\sigma, \varphi\, \mathcal{R}\, \psi) = \sup_{j\geq 0} \max\left\{V_4(\sigma_{j..}, \psi),\ \sup_{0\leq i<j} V_4(\sigma_{i..}, \varphi)\right\}.$$

11.5 Examples

In this section, we showcase the applicability of the proposed semantics with a number of examples.

11.5.1 The usefulness of implications that are not true

We argued in the previous section that rLTL captures the intended robustness properties for the specification $\boxdot p \Rrightarrow \boxdot q$ whenever this formula evaluates to 1111. But does the formula $\boxdot p \Rrightarrow \boxdot q$ still provide useful information when its value is lower than 1111? It follows from the semantics of implication that $V(\sigma, \boxdot p \Rrightarrow \boxdot q) = b$, for $b \prec 1111$, occurs when $V(\sigma, \boxdot q) = b$, i.e., whenever a value of b can be guaranteed despite b being smaller than $V(\sigma, \boxdot p)$. The value $V(\sigma, \boxdot p \Rrightarrow \boxdot q)$ thus describes which weakened guarantee follows from the environment assumption whenever the intended system guarantee does not. This can be seen as another measure of robustness: despite $\boxdot q$ not following from $\boxdot p$, the behavior of the system is not arbitrary, a value of b is still guaranteed. The usefulness that the different shades of false provide is further discussed in Section 14 in the context of a concrete case study.

11.5.2 GR(1) in rLTL

The GR(1) fragment of LTL is becoming increasingly popular for striking an interesting balance between its expressiveness and the complexity of the corresponding synthesis problem [BJP12]. A GR(1) formula is an LTL($\Box, \Diamond$) formula of the form:

$$\bigwedge_{i \in I} \Box\Diamond p_i \Rightarrow \bigwedge_{j \in J} \Box\Diamond q_j, \tag{11.24}$$

where p_i and q_j are atomic propositions and I, J are finite sets. We obtain the rLTL version of (11.24) simply by dotting the boxes and the diamonds:

$$\bigwedge_{i \in I} \boxdot \diamondsuit p_i \Rrightarrow \bigwedge_{j \in J} \boxdot \diamondsuit q_j. \tag{11.25}$$

Any valuation V for $\boxdot \diamondsuit p_i$ can be expressed in terms of a valuation W for LTL as:

$$V(\sigma, \boxdot \diamondsuit p_i) = (W(\sigma, \Box\diamondsuit p_i),\ W(\sigma, \Box\diamondsuit p_i),\ W(\sigma, \Box\diamondsuit p_i),\ W(\sigma, \diamondsuit p_i))\,.$$

Therefore, $V(\sigma, \boxdot \diamondsuit p_i)$ can only assume three different values: 1111 when $\Box\diamondsuit p_i$ holds, 0001 when $\Box\diamondsuit p_i$ fails to hold but $\diamondsuit p_i$ does hold, and 0000 when $\diamondsuit p_i$ fails to hold. Based on this observation, and assuming that (11.25) evaluates to 1111, we conclude that $\bigwedge_{j \in J} \Box\diamondsuit q_j$ holds whenever $\bigwedge_{i \in I} \Box\diamondsuit p_i$ does, as required by (11.24). In contrast to (11.24), however, the weakened system guarantee $\bigwedge_{j \in J} \diamondsuit q_j$ holds whenever the weaker environment assumption $\bigwedge_{i \in I} \diamondsuit p_i$ does.

11.5.3 Non-counting formulae

All the preceding examples were counting formulae. We now consider the simple non-counting formula $\Box(p \Rightarrow \diamondsuit q)$, which requires each occurrence of p to be followed by an occurrence of q. The word $\sigma_1 = \{p\}\{q\}\emptyset^\omega$ clearly satisfies this formula although its permutation $\sigma_2 = \{q\}\{p\}\emptyset^\omega$ does not. In addition to being a non-counting formula, $\Box(p \Rightarrow \diamondsuit q)$ is one of the most popular examples of an LTL formula used in the literature known as the "request-response" property, and, for this reason, constitutes a litmus test to rLTL. Actually, such formula is part of the reactive specification patterns identified in [DAC99], which are part of our case studies in Section 14. The semantics of the dotted version of $\Box(p \Rightarrow \diamondsuit q)$ can be expressed using an LTL valuation W as:

$$V(\sigma, \boxdot(p \Rrightarrow \diamondsuit q)) =$$
$$(W(\sigma, \Box(p \Rightarrow \diamondsuit q)), W(\sigma, \Box\diamondsuit p \Rightarrow \Box\diamondsuit q), W(\sigma, \diamondsuit\Box p \Rightarrow \Box\diamondsuit q), W(\sigma, \Box p \Rightarrow \diamondsuit q))\,.$$

It is interesting to observe how the semantics of $\varphi = \boxdot(p \Rrightarrow \diamondsuit q)$ recovers: strong fairness, also known as compassion, when the value of φ is 0111; weak fairness, also known as justice, when the value of φ is 0011; and the even weaker notion of fairness represented by the LTL formula $\Box p \Rightarrow \diamondsuit q$, when the value of φ is 0001. The fact that all these different and well known notions of fairness naturally appear in the proposed semantics is another strong indication of rLTL's naturalness and usefulness.

11.5.4 Counting with "robust release"

As we discussed before, the semantics of $\varphi \mathcal{R} \psi$ does not count how many times ψ holds before the first occurrence of φ. This property, however, is captured by the rLTL formula:

$$(\varphi \mathcal{R} \psi) \wedge (\neg\varphi \mathcal{U} \psi)\,. \tag{11.26}$$

To see why, we assume $\varphi = p$ and $\psi = q$ for atomic propositions p and q. Then, express the semantics of the rLTL formula (11.26) in terms of an LTL valuation W as:

$$V(\sigma, (p \mathcal{R} q) \wedge (\neg p \mathcal{U} q)) = (W(\sigma, p \mathcal{R} q), W(\neg p \mathcal{U} q), W(\neg p \mathcal{U} q), W(\neg p \mathcal{U} q))\,.$$

Note how we can now distinguish between three cases: (1) $p \mathcal{R} q$ holds, corresponding to value 1111; (2) q holds at least once before being released by p, corresponding to value 0111; and (3) q does not hold before being released by p, corresponding to value 0000.

11.5.5 Non-decomposition of "robust until"

The previous example showed how the LTL equivalence between $\varphi \mathcal{R} \psi$ and $(\varphi \mathcal{R} \psi) \wedge (\neg\varphi \mathcal{U} \psi)$ is not valid in rLTL. Another LTL equivalence that is not valid in rLTL is the decomposition of the until operator into its liveness and safety parts, that is the equivalence between $\varphi \mathcal{U} \psi$ and $\diamondsuit\psi \wedge (\psi \mathcal{R} (\psi \vee \varphi))$. The rLTL formula $\diamondsuit\psi \wedge (\psi \mathcal{R} (\psi \vee \varphi))$ expresses

a weaker requirement than $\varphi\,\mathcal{U}\,\psi$ that is also useful to express robustness. When φ and ψ are the atomic propositions p and q, respectively, the semantics of $\Diamonddot\psi \wedge (\psi\,\mathcal{R}\,(\psi \vee \varphi))$ can be expressed in terms of an LTL valuation W as:

$$V(\sigma, \Diamond q \wedge (q\,\mathcal{R}\,(q \vee p))) = (W(\sigma, p\,\mathcal{U}\,q),\ W(\sigma, \Diamond q),\ W(\sigma, \Diamond q),\ W(\sigma, \Diamond q))\,.$$

Whereas $\varphi\,\mathcal{U}\,\psi$ only assumes two values, $\Diamonddot\psi \wedge (\psi\,\mathcal{R}\,(\psi \vee \varphi))$ assumes three possible values allowing to separate the words that violate $\varphi\,\mathcal{U}\,\psi$ into those that satisfy $\Diamond q$ and those that do not.

CHAPTER 12

The rLTL model checking problem

12.1 Relating LTL and rLTL

In this section we discuss, at the technical level, the relationships between rLTL($\mathcal{P}$) and LTL($\mathcal{P}$). Recall the mappings $\pi_j : \mathbb{B}_5 \to \mathbb{B}$ introduced in (11.15), $\pi_j(a_1, a_2, a_3, a_4) = a_j$, $j \in \{1, 2, 3, 4\}$. Composing π_j with the rLTL valuation V we obtain the function $V_j = \pi_j \circ V$ that transforms an infinite word $\sigma \in \Sigma^\omega$ and an rLTL($\mathcal{P}$) formula φ into the element $V_j(\sigma, \varphi) \in \mathbb{B}$. Each truth value in $\mathbb{B}_5$ can be viewed as a sequence of 4 bits. Consequently, we show how to translate an rLTL($\mathcal{P}$) formula φ into four LTL($\mathcal{P}$) formulae $\varphi_1, \ldots, \varphi_4$ such that:

$$\pi_j(V(\sigma, \varphi)) = V_j(\sigma, \varphi) = W\left(\sigma, \varphi_j\right), \tag{12.1}$$

for all $\sigma \in \Sigma^\omega$ and $j \in \{1, \ldots, 4\}$. The key idea is to emulate the semantics of each operator occurring in φ component-wise by means of dedicated LTL formulae. To make this clear and straightforward, we define the operator:

$$\mathrm{ltl} : \{1, \ldots, 4\} \times \mathrm{rLTL}(\mathcal{P}) \to \mathrm{LTL}(\mathcal{P}), \tag{12.2}$$

as in Table 12.1. Then, each φ_j formula is constructed as:

$$\varphi_j \coloneqq \mathrm{ltl}(j, \varphi). \tag{12.3}$$

It is not hard to verify that the formulae φ_j have indeed the desired meaning. Moreover, notice that due to the ordering of the rLTL truth values in $\mathbb{B}_5$, see (11.6), it follows that:

$$W(\sigma, \varphi_j) \geq W(\sigma, \varphi_i), \text{ for } j \geq i. \tag{12.4}$$

Table 12.1: The rLTL semantics via the ltl operator.

Operator	Symbol	Semantics for $p \in \mathcal{P}$, $\varphi, \psi \in \text{rLTL}(\mathcal{P})$.
Atomic Proposition		$\forall i \in \{1,2,3,4\} : \text{ltl}(i, p) = p.$
Negation	$\neg$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \neg\varphi) = \neg\text{ltl}(1, \varphi).$
Conjunction	$\wedge$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \varphi \wedge \psi) = \text{ltl}(i, \varphi) \wedge \text{ltl}(i, \psi).$
Disjunction	$\vee$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \varphi \vee \psi) = \text{ltl}(i, \varphi) \vee \text{ltl}(i, \psi).$
Robust Implication	$\Rrightarrow$	$\forall i \in \{1,2,3\} : \text{ltl}(i, \varphi \Rrightarrow \psi) = (\text{ltl}(i, \varphi) \Rightarrow \text{ltl}(i, \psi)) \wedge \text{ltl}(i+1, \varphi \Rrightarrow \psi),$ $\text{ltl}(4, \varphi \Rrightarrow \psi) = (\text{ltl}(4, \varphi) \Rightarrow \text{ltl}(4, \psi)).$
Next	$\odot$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \odot\varphi) = \bigcirc\text{ltl}(i, \varphi).$
Robust Eventually	$\diamondsuit$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \diamondsuit\varphi) = \Diamond\text{ltl}(i, \varphi).$
Robust Always	$\boxdot$	$\text{ltl}(1, \boxdot\varphi) = \Box\text{ltl}(1, \varphi),$ $\text{ltl}(2, \boxdot\varphi) = \Diamond\Box\text{ltl}(2, \varphi),$ $\text{ltl}(3, \boxdot\varphi) = \Box\Diamond\text{ltl}(3, \varphi),$ $\text{ltl}(4, \boxdot\varphi) = \Diamond\text{ltl}(4, \varphi).$
Robust Until	$\mathcal{U}$	$\forall i \in \{1,2,3,4\} : \text{ltl}(i, \varphi \,\mathcal{U}\, \psi) = \text{ltl}(i, \varphi) \,\mathcal{U}\, \text{ltl}(i, \psi).$
Robust Release	$\mathcal{R}$	$\text{ltl}(1, \varphi \,\mathcal{R}\, \psi) = \text{ltl}(1, \varphi) \,\mathcal{R}\, \text{ltl}(1, \psi),$ $\text{ltl}(2, \varphi \,\mathcal{R}\, \psi) = \Diamond\text{ltl}(2, \varphi) \vee \Diamond\Box\text{ltl}(2, \psi),$ $\text{ltl}(3, \varphi \,\mathcal{R}\, \psi) = \Diamond\text{ltl}(3, \varphi) \vee \Box\Diamond\text{ltl}(3, \psi),$ $\text{ltl}(4, \varphi \,\mathcal{R}\, \psi) = \Diamond\text{ltl}(4, \varphi) \vee \Diamond\text{ltl}(4, \psi).$

Although the above construction only incurs a linear blowup in the number of subformulae of φ_j, the resulting LTL formula itself might be exponentially larger when explicitly constructed due to the recursive substitution.

Next, we show that the LTL($\mathcal{P}$) semantics can be recovered from the first bit of the rLTL($\mathcal{P}$) semantics. Specifically, for any $\phi \in \text{LTL}(\mathcal{P})$, there exists a $\varphi \in \text{rLTL}(\mathcal{P})$, such that $V_1(\sigma, \varphi) = W(\sigma, \phi)$ for every $\sigma \in \Sigma^\omega$. The construction of a suitable $\varphi \in \text{rLTL}(\mathcal{P})$, given any $\phi \in \text{LTL}(\mathcal{P})$, is straightforward given Table 12.1. First, every implication $\psi_1 \Rightarrow \psi_2$ in $\phi \in \text{LTL}(\mathcal{P})$ is replaced by the equivalent in LTL formula $\neg\psi_1 \vee \psi_2$. Second, ϕ is brought into negation normal form. Finally, by taking the robust version of the latter LTL formula, i.e., replacing the LTL temporal operators with their corresponding rLTL temporal operators, one obtains the desired $\varphi \in \text{rLTL}(\mathcal{P})$.

Remark 9. *In the construction above, we replace every implication in the LTL formula ϕ by its LTL-equivalent negation form. This is justified as, given the semantics of rLTL and LTL respectively, we have that for any infinite word $\sigma \in \Sigma^\omega$ the first bit of the rLTL valuation, $V_1(\sigma, \varphi \Rrightarrow \psi)$, and the LTL valuation, $W(\sigma, \varphi_1 \Rightarrow \psi_1)$, are related as follows:*

$$V_1(\sigma, \varphi \Rrightarrow \psi) \leq V_1(\sigma, \neg\varphi \vee \psi) = W(\sigma, \neg\varphi_1 \vee \psi_1) = W(\sigma, \varphi_1 \Rightarrow \psi_1).$$

Furthermore, we can construct simple examples for which the inequality above is strict, i.e., it fails to be an equality. For instance, consider the LTL formula $\Box p \Rightarrow \Box q$, its corresponding rLTL formula $\boxdot p \Rrightarrow \boxdot q$, and the word $\sigma = \emptyset\{p\}^\omega$. Then, on one hand it holds that $V_1(\sigma, \boxdot p \Rrightarrow \boxdot q) = 0$, but on the other hand we can easily verify the following equalities $V_1(\sigma, \neg \boxdot p \vee \boxdot q) = 1 = W(\sigma, \Box p \Rightarrow \Box q)$.

The preceding discussion leads to the following result.

Lemma 12.1.1. *Any rLTL formula $\varphi \in \text{rLTL}(\mathcal{P})$ can be translated to four LTL formulae $\varphi_j \in \text{LTL}(\mathcal{P})$, $j = 1, 2, 3, 4$, as in (12.3) and such that (12.1) holds, with $|\varphi_j| \leq c|\varphi|$, for some $c > 0$. Moreover, for any LTL formula $\phi \in \text{LTL}(\mathcal{P})$, one can systematically construct an rLTL formula $\varphi \in \text{rLTL}(\mathcal{P})$, such that $V_1(\sigma, \varphi) = W(\sigma, \phi)$ for every $\sigma \in \Sigma^\omega$ and with $|\varphi| = |\phi|$. The aforementioned translations imply decidability of any problem for rLTL($\mathcal{P}$), whose corresponding problem for LTL($\mathcal{P}$) is decidable. Moreover, due to their effective translations to each other, LTL($\mathcal{P}$) and rLTL($\mathcal{P}$) are equally expressive.*

Proof. Translating $\phi \in \text{LTL}(\mathcal{P})$ to $\varphi \in \text{rLTL}(\mathcal{P})$ only involves replacing the LTL temporal operators with their rLTL counterparts, which implies that $|\varphi| = |\phi|$. The translation of $\varphi \in \text{rLTL}(\mathcal{P})$ to $\varphi_j \in \text{LTL}(\mathcal{P})$, $j = 1, 2, 3, 4$, is provided in (12.3). This translation also results in a linear increase in the number of the unique subformulae, i.e., $|\varphi_j| \leq c|\varphi|$, and the exact coefficient $c > 0$ is computed in Appendix A.1. □

12.2 rLTL model-checking: definitions and background

Similarly to LTL, rLTL gives rise to various decision problems. One of the most important problems is the *model-checking* problem. In this section we first formalize this problem in the context of rLTL. Then, through the translation from rLTL to LTL provided in Section 12.1, the decidability of the rLTL model-checking problem is immediately settled. However, the treatment provided by the translation results in the construction of relatively large automata, which is inefficient for the purposes of model-checking. Therefore, we review the tighter known complexity bounds for the reader's convenience. Although the proof of such results is outside of the scope of this work (the interested reader is referred to [TN15, TN16] for more details), they provide the setting in which we can appreciate the results in Section 13 on an rLTL fragment for which the model checking problem can be solved with lower complexity.

As discussed in our review of LTL, given a model of a system that is represented as a GBA, the LTL model-checking problem essentially asks whether or not all possible computations of the model satisfy an LTL specification. In a similar manner, the *rLTL model-checking problem* is intuitively understood as the question of "to what degree does a model satisfy a specification?". The specification in this section is represented by an rLTL formula. For simplicity, we consider the model of the system to be given directly as a GBA. This leads to the following formulation of the *rLTL model-checking problem.*

Problem 5 (rLTL model-checking)**.** *Given a set of atomic propositions $\mathcal{P}$, a GBA $\mathcal{G}$ with the corresponding set of words $L(\mathcal{G}) \subseteq \left(2^{\mathcal{P}}\right)^{\omega}$ that it recognizes, an rLTL formula $\varphi \in \text{rLTL}(\mathcal{P})$,*

and a truth value $b \in \mathbb{B}_5$, we ask if $V(\sigma, \varphi) \geq b$ holds for all $\sigma \in L(\mathcal{G})$.

In practice, one would be more interested in finding what is the largest $b \in \mathbb{B}_5$ such that $V(\sigma, \varphi) \geq b$ holds for all $\sigma \in L(\mathcal{G})$. Section 12.3 provides the answer to Problem 5, and repeatedly solving this problem for decreasing values of b addresses the optimization problem of finding the largest $b \in \mathbb{B}_5$ such that $V(\sigma, \varphi) \geq b$ for all $\sigma \in L(\mathcal{G})$. Note that this requires at most four invocations of the rLTL model-checking procedure and, hence, does not change the asymptotic complexity of the problem.

12.3 Improved bounds for rLTL model-checking

In the preceding section, we provided a simple means to translate rLTL formulae into LTL formulae via the ltl operator in (12.2). Hence, as a consequence of Lemma 12.1.1 the decidability question for Problem 5 is settled. In practice, however, this translation involves a blow-up, which results in relatively large automata that would then be used to solve the rLTL model-checking problem.

To alleviate this problem, a more efficient approach, via a direct translation from rLTL($\mathcal{P}$) formulae into Generalized Büchi Automata (GBAs), was presented in [TN16] for the $\text{rLTL}_{\boxdot,\diamond}$ fragment and for the full rLTL in [TN15]. This construction, given an rLTL($\mathcal{P}$) formula φ, results in a GBA with $\mathcal{O}\left(5^{|\varphi|}\right)$ states, where $|\varphi|$ is the number of subformulae of φ. This is the same complexity as for the LTL translation, which results in an automaton with size in $\mathcal{O}\left(2^{|\varphi|}\right)$, once we replace 2 with 5 since rLTL is 5-valued while LTL is 2-valued. Recall that in this work, we reason about the model-checking complexity by focusing on the size of the corresponding GBA constructed from a given formula. On these grounds, we recall the following result from [TN16, TN15].

Theorem 12.3.1. *Given an* rLTL($\mathcal{P}$) *formula φ, the rLTL model-checking problem (Prob-*

lem 5) can be decided by using an automaton with at most:

$$\mathcal{O}\left(5^{|\varphi|}\right) \tag{12.5}$$

states, where $|\varphi|$ denotes the length of formula φ, i.e., the number of its distinct subformulae.

Interestingly, the many valued semantics of rLTL allows posing useful optimization problems in system verification. For instance, as discussed in the beginning of this section, one might be interested in the largest value that a system guarantees. Therefore, by repeatedly solving Problem 5 for decreasing truth values, one finds the largest value that a system guarantees. Theorem 12.3.1 provides a non-trivial upper bound, when compared to the direct translation via the ltl operator, as the following example suggests.

Example 1. *Consider the* rLTL($\mathcal{P}$) *formula* $\boxdot p \Rrightarrow \boxdot q$*, and assume we wish to check* $V(\sigma, \varphi) = 0111$ *for every* $\sigma \in L(\mathcal{G})$*, where* $\mathcal{G}$ *is the GBA representing the model of a given system. By using the* ltl *operator in Table 12.1, this is equivalent to model-checking the formula:*

$$\mathrm{ltl}(2, \boxdot p \Rrightarrow \boxdot q) = (\Box\Diamond p \Rightarrow \Box\Diamond q) \wedge (\Diamond\Box p \Rightarrow \Diamond\Box q) \wedge (\Diamond p \Rightarrow \Diamond q).$$

The original rLTL formula is of length 5*, and the LTL formula above has length* 15*. Hence, by naïvely applying LTL complexity bounds* (10.1)*, we count an upper bound of* 2^{15} *states the corresponding GBA, which is worse than the upper bound of* 5^5 *states from* (12.5)*.*

CHAPTER 13

A fragment for efficient rLTL model-checking

Theorem 12.3.1 states that the rLTL($\mathcal{P}$) model-checking problem can be solved by using a GBA of size $\mathcal{O}\left(5^{|\varphi|}\right)$. Nonetheless, this bound is still more expensive than the one of the LTL($\mathcal{P}$) model-checking problem, which is $\mathcal{O}\left(2^{|\varphi|}\right)$. It is then natural to ask the following question, can we find a fragment of rLTL($\mathcal{P}$) for which the model-checking problem can be solved more efficiently?

To motivate an answer to the above question, we show that the high complexity of rLTL($\mathcal{P}$) model-checking stems from the fact that the four bits of an rLTL truth value are *coupled* by the $\Rrightarrow$ and $\neg$ operators.

Example 2. *Consider the* rLTL($\mathcal{P}$) *formula* φ *given by* $\neg(p \Rrightarrow (q \mathcal{R} r))$*, where* p*,* q *and* r *are atomic propositions. To compute, for example, the* 4*-th bit of its valuation, one needs to unfold the corresponding* LTL($\mathcal{P}$) *formula:*

$$\begin{aligned}
\mathrm{ltl}(4, \neg(p \Rrightarrow (q \mathcal{R} r))) &= \neg \mathrm{ltl}(1, p \Rrightarrow (q \mathcal{R} r)) \\
&= \neg\Big((\mathrm{ltl}(1,p) \Rightarrow \mathrm{ltl}(1, q \mathcal{R} r)) \wedge \mathrm{ltl}(2, p \Rrightarrow (q \mathcal{R} r)\Big) \\
&= \neg\Big((p \Rightarrow (q \mathcal{R} r)) \wedge \mathrm{ltl}(2, (p \Rrightarrow (q \mathcal{R} r)))\Big) \\
&= \big(p \wedge \neg(q \mathcal{R} r)\big) \vee \neg\Big(\big(\mathrm{ltl}(2,p) \Rightarrow \mathrm{ltl}(2, q \mathcal{R} r)\big) \wedge \mathrm{ltl}(3, p \Rrightarrow (q \mathcal{R} r))\Big) \\
&= \big(p \wedge \neg(q \mathcal{R} r)\big) \vee \neg\Big(\big(p \Rightarrow (\Diamond q \vee \Diamond\Box r)\big) \wedge \mathrm{ltl}(3, p \Rrightarrow (q \mathcal{R} r))\Big).
\end{aligned}$$

If we continue unfolding the formula, we see that one needs to check an LTL formula that non-trivially depends on the bits 1, 2, and 3 of the valuation of φ*.*

With this intuition in mind, we aim to identify a fragment of rLTL($\mathcal{P}$) that suffers minimally from the coupling required by, e.g., the robust implication.

13.1 Main results

We begin by defining the following fragments of rLTL($\mathcal{P}$).

Definition 19 (Fragments $\mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ and $\overline{\mathrm{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$)**.** *Given a set of atomic propositions $\mathcal{P}$, define $\mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P}) \subset \mathrm{rLTL}(\mathcal{P})$ as the set of all rLTL formulae that either do not contain any $\Rrightarrow$ operator, or if they do, the antecedent of the implication does not contain any $\boxdot$ or $\mathcal{R}$ operators. Then define the more general fragment $\overline{\mathrm{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ as:*

$$\overline{\mathrm{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P}) = \mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P}) \bigcup \left\{ \psi_1 \Rrightarrow \psi_2 \middle| \psi_1, \psi_2 \in \mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P}) \right\}, \tag{13.1}$$

which allows for one $\Rrightarrow$ operator on the outermost level with no restrictions on the implication's antecedent.

The main result of this section establishes refined complexity bounds for the model-checking problem of the above defined fragments.

Theorem 13.1.1. *Consider a set of atomic propositions $\mathcal{P}$. The rLTL model-checking problem for any formula in $\overline{\mathrm{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ can be solved by performing at most 4 LTL model-checking steps, each using an automaton with at most:*

$$\mathcal{O}\left(2^{|\varphi|-\kappa(\varphi)} 3^{\kappa(\varphi)}\right) \tag{13.2}$$

states, where $\kappa(\varphi) = \mathrm{card}\left(\{\psi \in \mathrm{cl}(\varphi) \mid \psi = \boxdot\psi_1\}\right) + \mathrm{card}\left(\{\psi \in \mathrm{cl}(\varphi) \mid \psi = \psi_1 \,\mathcal{R}\, \psi_2\}\right)$, i.e., the number of distinct subformulae of φ of the form $\boxdot\psi$ and $\psi_1 \,\mathcal{R}\, \psi_2$, and $|\varphi|$ is the length of φ.

Remark 10. *Similarly to LTL, one can verify that for any $\varphi \in \mathrm{rLTL}(\mathcal{P})$, it holds that $\diamondsuit\varphi = true\ \mathcal{U}\ \varphi$, and $\boxdot\varphi = false\ \mathcal{R}\ \varphi$. Hence, the function $\kappa(\varphi)$ can be understood as*

the number of distinct $\psi_1 \mathcal{R} \psi_2$ subformulae of φ, accounting for those underlying the $\boxdot$ operators.

The above result provides a smaller complexity bound for the rLTL model-checking problem, which is closer to the tight bound of the LTL model-checking problem, i.e., $\mathcal{O}\left(2^{|\varphi|}\right)$. In fact, in the absence of $\boxdot$ and $\mathcal{R}$ operators, the bound in (13.2) reduces to that of LTL. A special case of Theorem 13.1.1 is the result proposed in [APN18], where the same complexity bound is proven for a smaller rLTL fragment, specifically the set of rLTL formulae that do not contain any $\mathcal{R}$ or $\Rrightarrow$ (regardless of their antecedent) operators, or that allow for at most one $\Rrightarrow$ operator only on the outermost level.

Remark 11. *Throughout this section we compare the model-checking complexity of the proposed fragment $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ to that of full LTL. This is justified by observing that for any LTL formula $\phi \in \text{LTL}(\mathcal{P})$, after replacing every implication $\psi_1 \Rightarrow \psi_2$ therein by $\neg\psi_1 \vee \psi_2$, the corresponding rLTL formula belongs to the proposed fragment. This observation follows directly by* (13.1).

To illustrate the practicality of the proposed fragment $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$, we examine the LTL formulae found in the Büchi Store [TTC13], an open repository of LTL formulae. More than 95% of the corresponding rLTL versions of these formulae belong to $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$. Moreover, as discussed in Section 14.3 in more detail, all the relevant reactivity patterns [DAC99] fit into the fragment $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$. Finally, the proposed fragment includes the GR(1) fragment. These considerations establish $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ as a practically useful fragment of rLTL.

We devote the rest of this section to formally proving Theorem 13.1.1. Towards this, we introduce Algorithm 1. At a high level, Algorithm 1 translates an rLTL formula into four LTL formulae and then model-checks each of them. The key idea is that when operating within the proposed fragment, the four formulae are independent of each other. It is actually this independence that allows us to construct smaller automata, by the use of temporal testers, and attain the desired complexity bounds. By exploiting the ordering of the five truth values

Algorithm 1: Model-checking algorithm for the $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ fragment.

Input: A language $L(\mathcal{G})$ generated by a GBA $\mathcal{G}$, and a formula $\varphi \in \overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$.

Output: The truth value of φ on $L(\mathcal{G})$, i.e., $b(L(\mathcal{G}), \varphi) \in \mathbb{B}_5$.

for $j = 0, 1, 2, 3$ **do**
  $w := \inf_{\sigma \in L(\mathcal{G})} W(\sigma, \text{ltl}(4 - j, \varphi))$.
  if $w = 0$ **then**
    return $\mathbb{B}_5[j]$
return $\mathbb{B}_5[4]$

in (11.6) the algorithm stops once an LTL formula is falsified.

Overall in this section we prove the following points: (1) the fragment $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ allows independent bit-wise computations for the rLTL model-checking problem; (2) smaller automata can be constructed for the formulae in the $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ fragment by utilizing temporal testers; and (3) for any formula in the $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ fragment, the claimed bound on the size of the corresponding automaton is satisfied.

13.2 Bit-wise independence of $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ and insights on the robust implication

The goal of this section is to establish that *for any formula* $\varphi \in \text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$*, the truth value of the* j*-th bit of its valuation,* $V_j(\sigma, \varphi)$*, is independent of any bit* $i \neq j$. Recall from Definition 19 that $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ contains all rLTL formulae that either do not contain any $\Rightarrow$ operators or, if they do, there exist no $\boxdot$ or $\mathcal{R}$ operators in the implication's antecedent.

For the formulae that do not contain any $\Rightarrow$ operators, the desired result is easily verified by inspection of Table 12.1. For the formulae that contain $\Rightarrow$, but their assumptions do not contain any $\boxdot$ or $\mathcal{R}$ operators we perform the following analysis.

As Examples 1 and 2 show, the $\Rightarrow$ operator induces coupling of the different LTL formulae

corresponding to the bits of an rLTL truth value. Thus, we now draw insight on when it makes sense for an implication to be evaluated robustly. The notion of robustness, namely that "small violations of the assumption lead to, at most, small violations of the guarantee", is infused in the semantics of the $\Rrightarrow$ operator, see (11.13), since by *weakening* the assumption and the guarantee, different rLTL truth values are obtained. Consequently, we investigate what "weakening a formula" means in the context of rLTL.

Definition 20 (Weakened rLTL($\mathcal{P}$) formula)**.** *Given a formula $\varphi \in \text{rLTL}(\mathcal{P})$, we say that φ admits a* weakened version *if there exists an infinite word $\sigma \in \Sigma^\omega$ such that the valuation of φ on σ we have that $V(\varphi, \sigma) \in \{0001, 0011, 0111\}$.*

Based on the above definition, for any rLTL formula φ that does not admit a weakened version and any infinite word $\sigma \in \Sigma^\omega$, we have that $V(\varphi, \sigma) \in \{0000, 1111\}$, i.e., the valuation of φ admits only a binary truth value. This is equivalent to the statement that the corresponding LTL formulae φ_j, for $j \in \{1, 2, 3, 4\}$, defined in (12.3) are semantically equivalent. Given the rLTL semantics, we make the following crucial observation.

Proposition 13.2.1. *Given a formula $\varphi \in \text{rLTL}(\mathcal{P})$, if φ admits a* weakened version*, then φ contains at least one $\boxdot$ or one $\mathcal{R}$ operator.*

Proof. We prove the result by contraposition. Consider any $\varphi \in \text{rLTL}(\mathcal{P})$ that does not contain any $\boxdot$ or $\mathcal{R}$ operators. The proof proceeds in three steps.

1. If, additionally, φ does not contain any $\Rrightarrow$ operators, by inspection of Table 12.1 we see that the corresponding LTL formulae φ_j, for $j \in \{1, 2, 3, 4\}$, defined in (12.3) are identical. Thus, the valuations $W(\sigma, \varphi_j)$, $j \in \{1, 2, 3, 4\}$, over any $\sigma \in \Sigma^\omega$ are equal, and, hence, $V(\sigma, \varphi) \in \{0000, 1111\}$.

2. If φ is of the form $\phi \Rrightarrow \psi$, where neither ϕ nor ψ contain a $\Rrightarrow$ operator, then from the semantics of robust implication in (11.13) we have that $V(\sigma, \varphi) = 1111$ if $V(\sigma, \phi) \preceq$

$V(\sigma, \psi)$, and that $V(\sigma, \varphi) = V(\sigma, \psi)$ otherwise. Furthermore, from the point above $V(\sigma, \phi), V(\sigma, \psi) \in \{0000, 1111\}$, and, hence, $V(\varphi, \sigma) \in \{0000, 1111\}$.

3. Finally, if $\varphi \in \text{rLTL}(\mathcal{P})$ does not contain any $\boxdot$ or $\mathcal{R}$ operators, but may contain an arbitrary number of $\Rrightarrow$ operators, the same result follows simply by induction on the subformulae of φ.

Therefore, if φ does not contain a $\boxdot$, or a $\mathcal{R}$ operator, then it does not admit a *weakened version*. Equivalently, if φ admits a weakened version, then φ contains at least one $\boxdot$, or one $\mathcal{R}$ operator. This concludes the proof. □

We use this proposition to determine when it is necessary to evaluate an implication robustly. Given an rLTL formula of the form $\varphi \Rrightarrow \psi$, if the assumption φ does not admit a weakened version, then the implication does not have to be evaluated robustly, and the LTL equivalence $\neg\varphi \vee \psi$ can be used instead. The following proposition formalizes this idea.

Proposition 13.2.2. *An* rLTL($\mathcal{P}$) *formula* $\varphi \Rrightarrow \psi$ *is* semantically equivalent *to* $\neg\varphi \vee \psi$*, i.e., for any* $\sigma \in \Sigma^\omega$ *it holds that* $V(\sigma, \varphi \Rrightarrow \psi) = V(\sigma, \neg\varphi \vee \psi)$ *if* φ *does not contain any* $\boxdot$ *or* $\mathcal{R}$ *operators.*

Proof. Consider the rLTL formula $\varphi \Rrightarrow \psi$, where φ does not contain a robust implication operator for simplicity. The valuation of $\varphi \Rrightarrow \psi$ over $\sigma \in \Sigma^\omega$, denoted by $V(\sigma, \varphi \Rrightarrow \psi)$, is equal to:

$$\left(W\left(\sigma, \bigwedge_{j=1}^{4}(\varphi_j \Rightarrow \psi_j)\right), W\left(\sigma, \bigwedge_{j=2}^{4}(\varphi_j \Rightarrow \psi_j)\right), W\left(\sigma, \bigwedge_{j=3}^{4}(\varphi_j \Rightarrow \psi_j)\right), W\left(\sigma, \varphi_4 \Rightarrow \psi_4\right)\right), \tag{13.3}$$

where φ_j, ψ_j, for $j \in \{1, 2, 3, 4\}$, are defined in (12.3) by using the ltl operator according to Table 12.1. If φ contains no $\boxdot$ and no $\mathcal{R}$, as described by Proposition 13.2.1, the LTL

formulae φ_j are identical. Let us denote all of them by φ_1. Then $V(\sigma, \varphi \Rrightarrow \psi)$ becomes:

$$\left(W\left(\sigma, \bigwedge_{j=1}^{4}(\varphi_1 \Rightarrow \psi_j)\right), W\left(\sigma, \bigwedge_{j=2}^{4}(\varphi_1 \Rightarrow \psi_j)\right), W\left(\sigma, \bigwedge_{j=3}^{4}(\varphi_1 \Rightarrow \psi_j)\right), W\left(\sigma, \varphi_1 \Rightarrow \psi_4\right)\right), \tag{13.4}$$

where we can see that, contrary to (13.3), all the antecedents are now identical. We now show that (13.4) is equal to:

$$\left(W\left(\sigma, \varphi_1 \Rightarrow \psi_1\right), W\left(\sigma, \varphi_1 \Rightarrow \psi_2\right), W\left(\sigma, \varphi_1 \Rightarrow \psi_3\right), W\left(\sigma, \varphi_1 \Rightarrow \psi_4\right)\right). \tag{13.5}$$

For any $\sigma \in \Sigma^\omega$:

1. Assume $W(\sigma, \varphi_1) = 0$. Then $W(\sigma, \varphi_1 \Rightarrow \psi_j) = 1$, for $j \in \{1, 2, 3, 4\}$, and both (13.4) and (13.5) are equal to 1111.

2. Assume $W(\sigma, \varphi_1) = 1$, and denote each truth value in $\mathbb{B}_5$ as:

$$\mathbb{B}_5 = \{0000, 0001, 0011, 0111, 1111\} = \{\mathbb{B}_5[0], \mathbb{B}_5[1], \mathbb{B}_5[2], \mathbb{B}_5[3], \mathbb{B}_5[4]\}.$$

   If $W(\sigma, \psi_j) = 0$ for all $j \in \{1, 2, 3, 4\}$, then both (13.4) and (13.5) are equal to $\mathbb{B}_5[0]$. Else, let k be the smallest element of $\{1, 2, 3, 4\}$ such that $W(\sigma, \psi_k) = 1$. Then, by (12.4), we have that $W(\sigma, \psi_j) \geq W(\sigma, \psi_k)$ for $j \geq k$, yielding $W(\sigma, \psi_j) = 1$ for $j \geq k$, which, in turn, implies that both (13.4) and (13.5) are equal to $\mathbb{B}_5[5 - k]$.

The above shows that (13.4) and (13.5) are equal when evaluated over any $\sigma \in \Sigma^\omega$. By using the LTL equivalence between $\varphi_j \Rightarrow \psi_j$ and $\neg\varphi_j \vee \psi_j$, we have that (13.5) is equal to:

$$\left(W\left(\sigma, \neg\varphi_1 \vee \psi_1\right), W\left(\sigma, \neg\varphi_1 \vee \psi_2\right), W\left(\sigma, \neg\varphi_1 \vee \psi_3\right), W\left(\sigma, \neg\varphi_1 \vee \psi_4\right)\right), \tag{13.6}$$

and, thus, equal to (13.4). This yields the desired equality $V(\sigma, \varphi \Rrightarrow \psi) = V(\sigma, \neg\varphi \vee \psi)$.

The result for any $\varphi \in \text{rLTL}(\mathcal{P})$ that does not contain any $\boxdot$, or $\mathcal{R}$ operators, but may contain an arbitrary number of $\Rrightarrow$ operators, follows simply by induction on the subformulae of φ since neither the assumptions, nor the guarantees of these implications contain any $\boxdot$ or $\mathcal{R}$ operators. $\square$

Finally, by Proposition 13.2.2 and the rLTL semantics in Table 12.1, we are able to conclude that for any $\varphi \in \mathrm{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ the value of $V_j(\sigma, \varphi)$ is independent of any bit $i \neq j$. This concludes the objective of this section.

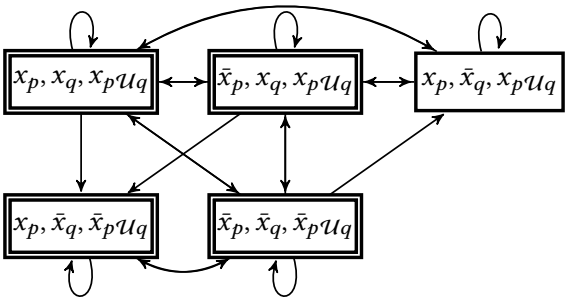
13.3 Introducing temporal testers

In this section, we review the concept of temporal testers [PZ08, KPR98]. Temporal testers are discrete transition systems equipped with justice conditions. One of the appeals of temporal testers is that they can be used to obtain automata recognizing infinite words that satisfy an LTL formula by composing testers recognizing its subformulae. For example, from testers for the formulae $p \,\mathcal{U}\, q$ and $\Box r$, one can construct a tester for $p \,\mathcal{U}\, (\Box r)$ by composing the testers for $p \,\mathcal{U}\, q$ and $\Box r$ using the constraint $q = \Box r$. We start by providing the necessary definitions.

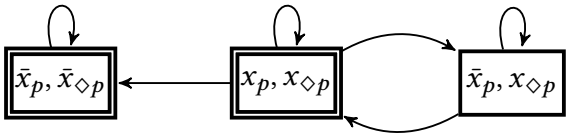
Definition 21. *A temporal tester for an LTL formula $\varphi \in \mathrm{LTL}(\mathcal{P})$ is defined as a tuple $\mathrm{T}_\varphi = (S, \Theta, R, \mathcal{J})$ where:*

- *S is the set of states, $S \subseteq \mathbb{B}^{\mathrm{cl}(\varphi)}$. Each state $x \in S$ is a function $x : \mathrm{cl}(\varphi) \to \mathbb{B}$ mapping a formula $\psi \in \mathrm{cl}(\varphi)$ to an element of $\mathbb{B}$. We denote the evaluation of x on ψ as x_ψ and interpret it as the truth value of ψ at the state x.*
- *$\Theta \subseteq S$ is a set of initial states.*
- *$R \subseteq S \times S$ is a transition relation.*
- *$\mathcal{J} = \{J_1, \ldots, J_K\} \subseteq 2^S$ is the set of justice requirements, where $J \subseteq S$ for each $J \in \mathcal{J}$.*

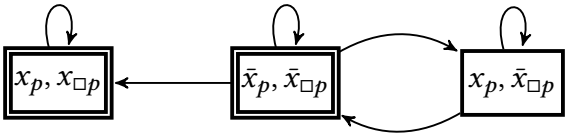
A computation *of a tester is an infinite sequence of states $\gamma = x^{(0)}x^{(1)}\ldots$ such that $x^{(0)} \in \Theta$, $\left(x^{(i)}, x^{(i+1)}\right) \in R$ for $i \geq 0$, and for every $J \in \mathcal{J}$, γ contains infinitely many states $x^{(i)} \in J$. Given a computation γ, we let $\sigma(\gamma) \in \left(2^{\mathcal{P}}\right)^\omega$ be the word $\sigma(\gamma) = \sigma_0(\gamma)\sigma_1(\gamma)\ldots$ where $\sigma_i(\gamma)$ is the subset of $\mathcal{P}$ defined by $p \in \sigma_i(\gamma)$ if and only if $x_p^{(t)} = 1$.*



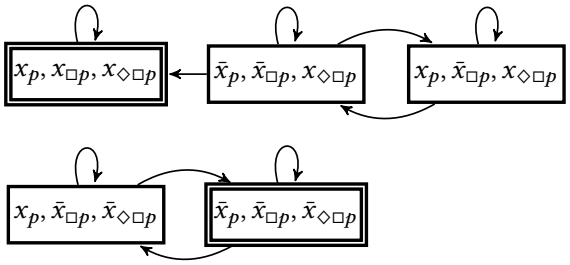
(a) Tester $\mathrm{T}_{p\mathcal{U}q}$.



(b) Tester $\mathrm{T}_{\diamond p}$.



(c) Tester $\mathrm{T}_{\Box p}$.



(d) Tester $\mathrm{T}_{\diamond \Box p}$ with two disjoint components.

Figure 13.1: At each state, for a subformula ψ, x_ψ denotes $x_\psi = 1$, and $\bar{x}_\psi$ denotes $x_\psi = 0$. All states are valid initial states. The double line states are contained in the justice requirement.

It is easily verified that the above definition of a temporal tester is equivalent to the one in [PZ08, KPR98].

Example 3 (A tester for the until operator adapted from [PZ08])**.** *A tester for $p\,\mathcal{U}\,q$, where p and q are atomic propositions, is as follows:*

$$\mathrm{T}_{p\mathcal{U}q} : \begin{cases} S = \left\{x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}\right\}, \\ \Theta = S, \\ R = \left\{(x, x') \in S \times S \mid x_{p\mathcal{U}q} = x_q \vee \left(x_p \wedge x'_{p\mathcal{U}q}\right)\right\}, \\ \mathcal{J} = \{J_1\},\ J_1 = \{x \in S \mid \neg x_{p\mathcal{U}q} \vee x_q = 1\}. \end{cases} \tag{13.7}$$

The above tester can be represented as an automaton with 5 states*, see Figure 13.1a.*

In the general case, the tester $\mathrm{T}_{\varphi\mathcal{U}\psi}$, where φ and ψ are LTL($\mathcal{P}$) formulae, is constructed

by *composition* of the testers for their subformulae. The following definition is adapted from [KPR98, Section 3.2], [PZ08, Section 7].

Definition 22 (Composition of Temporal Testers)**.** *The synchronous parallel composition of two testers is* $(S, \Theta, R, \mathcal{J}) = (S_1, \Theta_1, R_1, \mathcal{J}_1) \mid\mid\mid (S_2, \Theta_2, R_2, \mathcal{J}_2)$*, where* $S = S_1 \cup S_2$*,* $\Theta = \Theta_1 \cap \Theta_2$*,* $R = R_1 \cap R_2$*, and* $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2$*.*

1. *For a unary LTL operator* op*, a tester* $\mathrm{T}_{\mathrm{op}(\varphi)}$ *is given by* $\mathrm{T}_{\mathrm{op}(p)|p\leftarrow\varphi} \mid\mid\mid \mathrm{T}_{\varphi}$*, where the subscript* $p \leftarrow \varphi$ *denotes that we replace every instance of* x_p *and* $x_{\mathrm{op}(p)}$ *in the first tester, by* x_φ *and* $x_{\mathrm{op}(\varphi)}$ *respectively.*

2. *For a binary LTL operator* op*, a tester* $\mathrm{T}_{\mathrm{op}(\varphi_1,\varphi_2)}$ *is given by* $\mathrm{T}_{\mathrm{op}(p,q)|p\leftarrow\varphi_1,q\leftarrow\varphi_2} \mid\mid\mid \mathrm{T}_{\varphi_1} \mid\mid\mid \mathrm{T}_{\varphi_2}$*, where we replace every instance of* x_p*,* x_q *and* $x_{\mathrm{op}(p,q)}$ *in the first tester, by* x_{φ_1}*,* x_{φ_2}*, and* $x_{\mathrm{op}(\varphi_1,\varphi_2)}$ *respectively.*

By using the identities $\Diamond p = \mathit{true}\ \mathcal{U}\ p$, $\Box p = \neg\Diamond\neg p$, and $\mathrm{T}_{p\mathcal{U}q}$, we construct $\mathrm{T}_{\Diamond p}$ and $\mathrm{T}_{\Box p}$, which are shown in Figures 13.1b and 13.1c. By composing them, we obtain $\mathrm{T}_{\Diamond\Box p}$, shown in Figure 13.1d. Such testers play an important role in proving smaller upper bounds for the rLTL model-checking problem. Towards this, the following results from [APN18] provide recursive bounds on the size of a tester T_φ for an LTL($\mathcal{P}$) formula φ.

Definition 23 (Size of a tester)**.** *Given a tester* T_φ *for* $\varphi \in \mathrm{LTL}(\mathcal{P})$*, let* $|\mathrm{T}_\varphi|$ *denote its size, i.e., the number of its states, and let* $|\mathrm{T}_\varphi|_i$ *be the number of states where* $x_\varphi = i$*. Then, for any formulae* $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$*, and* $i, j, k \in \mathbb{B}$*:*

- *for any unary operator* op*,* $|\mathrm{T}_{\mathrm{op}(\varphi)}|_{i,j}$ *is the number of states where* $x_\varphi = i, x_{\mathrm{op}(\varphi)} = j$*,*
- *for any binary operator* op*,* $|\mathrm{T}_{\mathrm{op}(\varphi_1,\varphi_2)}|_{i,j,k}$ *is the number of states where* $x_\varphi = i$*,* $x_\psi = j$*,* $x_{\mathrm{op}(\varphi,\psi)} = k$*.*

The number of states in a tester can be decomposed as follows for any $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$:

$$|\mathrm{T}_{\mathrm{op}(\varphi)}| = \sum_{i,j} |\mathrm{T}_{\mathrm{op}(\varphi)}|_{i,j}, \quad |\mathrm{T}_{\mathrm{op}(\varphi_1,\varphi_2)}| = \sum_{i,j,k} |\mathrm{T}_{\mathrm{op}(\varphi_1,\varphi_2)}|_{i,j,k}.$$

Proposition 13.3.1. *Let p, q be two atomic propositions in $\mathcal{P}$, $\psi_1, \psi_2 \in \mathrm{LTL}(\mathcal{P})$ be two LTL formulae, and* op *denote an LTL operator. The following holds:*

$$|\mathrm{T}_{\mathrm{op}(\psi_1)}|_{i,j} \leq |\mathrm{T}_{\psi_1}|_i \cdot |\mathrm{T}_{\mathrm{op}(p)}|_{i,j}, \tag{13.8}$$

$$|\mathrm{T}_{\mathrm{op}(\psi_1,\psi_2)}|_{i,j,k} \leq |\mathrm{T}_{\psi_1}|_i \cdot |\mathrm{T}_{\psi_2}|_j \cdot |\mathrm{T}_{\mathrm{op}(p,q)}|_{i,j,k}. \tag{13.9}$$

Corollary 13.3.2 (Recursive Bounds)**.** *Consider a tester T_φ for $\varphi \in \mathrm{LTL}(\mathcal{P})$. The following recursive bounds hold on its number of states $|\mathrm{T}_\varphi|$:*

$$\textit{if } \varphi \textit{ is } p \in \mathcal{P} : |\mathrm{T}_\varphi| = 2, \tag{13.10}$$

$$\textit{if } \varphi \textit{ is } \neg\psi : \forall i, j : |\mathrm{T}_{\neg\psi}|_{i,j} = \begin{cases} |\mathrm{T}_\psi|_i, & \textit{if } i \neq j, \\ 0 & \textit{otherwise}, \end{cases} \tag{13.11}$$

$$\textit{if } \varphi \textit{ is } \mathrm{op}(\psi),\ \mathrm{op} \in \{\Diamond, \Box, \bigcirc\} : |\mathrm{T}_\varphi| \leq 2 \cdot |\mathrm{T}_\psi|, \tag{13.12}$$

$$\textit{if } \varphi \textit{ is } \Diamond\Box\psi : |\mathrm{T}_\varphi| = |\mathrm{T}_{\Diamond\Box\psi}| \leq 3 \cdot |\mathrm{T}_\psi|, \tag{13.13}$$

$$\textit{if } \varphi \textit{ is } \mathrm{op}(\psi_1, \psi_2),\ \mathrm{op} \in \{\vee, \wedge, \Rightarrow\} : |\mathrm{T}_\varphi| \leq |\mathrm{T}_{\psi_1}| \cdot |\mathrm{T}_{\psi_2}|, \textit{ and} \tag{13.14}$$

$$\textit{if } \varphi \textit{ is } \psi_1 \,\mathcal{U}\, \psi_2 \textit{ or } \varphi \textit{ is } \psi_1 \,\mathcal{R}\, \psi_2 : |\mathrm{T}_\varphi| \leq 2 \cdot |\mathrm{T}_{\psi_1}| \cdot |\mathrm{T}_{\psi_2}|. \tag{13.15}$$

Similarly to LTL, see Corollary 10.0.2, the complexity of the rLTL model-checking problem is proportional to the size of the GBAs constructed, see Theorem 12.3.1. Hence, we are motivated to construct GBAs with the smallest possible number of states and accepting conditions. We focus on elementary temporal testers arising in the study of rLTL formulae and use them to construct smaller GBAs using the following remark.

Remark 12 (Link with Generalized Büchi Automata)**.** *For any tester $\mathrm{T}_\varphi = (S, \Theta, R, \mathcal{J})$, one can construct a GBA $\mathcal{G}_\varphi = (Q, \Sigma, Q_0, \Delta, \mathcal{F})$ whose runs correspond to the computations of the tester as follows:*

- *$Q = S$, $Q_0 = \{x \in \Theta \mid x_\varphi = 1\}$, and $\mathcal{F} = \mathcal{J}$.*
- *$(q, \sigma, q') \in \Delta$ if and only if $(q, q') \in R$ and $\sigma = \{p \in \mathcal{P} \mid q'_p = 1\}$.*

Notice the relation between Q_0 and Θ. Since T_φ detects whether a computation satisfies φ or $\neg\varphi$, in order to obtain a GBA $\mathcal{G}_\varphi$ whose runs satisfy φ, it suffices to remove from Θ any state x satisfying $x_\varphi = 0$.

Remark 13 (Justice requirements)**.** *In $\mathrm{T}_{p\mathcal{U}q}$, and therefore in $\mathrm{T}_{\Diamond p}$, $\mathrm{T}_{\Box p}$, the number of justice requirements $J \in \mathcal{J}$ is always 1. By Definition 22, it follows that the composition $\mathrm{T}_{\Diamond\Box p} = \mathrm{T}_{\Diamond p|p\leftarrow\Box p} \,|||\, \mathrm{T}_{\Box p}$ has two justice requirements:*

$$\mathcal{J} = \{x \in S \mid \neg x_{\Diamond\Box p} \vee x_{\Box p} = 1\} \cup \{x \in S \mid x_{\Box p} \vee \neg x_p = 1\}.$$

The above justice requirements are met simultaneously at two of the states, $(x_p, x_{\Box p}, x_{\Diamond\Box p})$ and $(\bar{x}_p, \bar{x}_{\Box p}, \bar{x}_{\Diamond\Box p})$. In light of this, we use $\mathcal{J} = \{x \in S \mid (x_p \wedge x_{\Box p} \wedge x_{\Diamond\Box p}) \vee \neg(x_p \vee x_{\Box p} \vee x_{\Diamond\Box p}) = 1\}$ for the tester in Figure 13.1d, while preserving the computations of $\mathrm{T}_{\Diamond\Box p} = \mathrm{T}_{\Diamond p|p\leftarrow\Box p} \,|||\, \mathrm{T}_{\Box p}$. In this regard, the tester $\mathrm{T}_{\Diamond\Box p}$ in Figure 13.1d is optimized.

We are now ready to proceed onto constructing smaller, specialized GBAs for the fragment $\mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ and prove the refined complexity upper bounds.

13.4 Refined complexity bounds

The next lemma is integral to providing the promised complexity bounds of Theorem 13.1.1.

Lemma 13.4.1. *Given a set of atomic propositions $\mathcal{P}$, for any $\varphi \in \mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ and any $j \in \{1, 2, 3, 4\}$:*

$$|\mathrm{T}_{\mathrm{ltl}(j,\varphi)}| \leq 2^{|\varphi|-\kappa(\varphi)} 3^{\kappa(\varphi)}, \tag{13.16}$$

where $\kappa(\varphi) = \mathrm{card}\left(\{\psi \in \mathrm{cl}(\varphi) \mid \psi = \boxdot\psi_1\}\right) + \mathrm{card}\left(\{\psi \in \mathrm{cl}(\varphi) \mid \psi = \psi_1 \,\mathcal{R}\, \psi_2\}\right)$, i.e., the number of distinct subformulae of φ of the form $\boxdot\psi$ and $\psi_1 \,\mathcal{R}\, \psi_2$, and $|\varphi|$ is the length of φ.

Proof. To maintain a streamlined presentation we provide a sketch of the proof here. The full proof of Lemma 13.4.1 is found in Appendix A.2.

Given any $\varphi \in \text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ we prove the claimed bound on $|\text{T}_{\text{ltl}(j,\varphi)}|$, $j \in \{1,2,3,4\}$, by induction on the length of the formula for all operators except for the $\mathcal{R}$ operator. In the base case where φ is of length 1, i.e., an atomic proposition, (13.16) is satisfied as an equality given (13.10). For the induction step, if φ is of the form $\text{op}(\psi)$, with op any unary rLTL operator, or of the form $\text{op}(\psi_1, \psi_2)$, with op any binary operator except for $\mathcal{R}$, then the claim in (13.16) is proved using (13.11) through (13.15).

Finally, in the case of the $\mathcal{R}$ operator we construct specialized testers for each bit, which we prove to be correct and to satisfy the claimed bound (13.16) in the Appendix. □

So far, Lemma 13.4.1 shows that we can construct specialized temporal testers of smaller size. These, in turn, can be used to construct smaller GBAs as per Remark 12. To conclude the proof of Theorem 13.1.1, we consider Algorithm 1 for rLTL model-checking. This algorithm model-checks the LTL formulae corresponding to each bit of the truth value of an rLTL formula, and by exploiting the order of the truth values in $\mathbb{B}_5$, that is $0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111$, it benefits from an early stopping criterion if the value of a bit is zero. In particular, consider the language $L(\mathcal{G})$ generated by a GBA $\mathcal{G}$, and assume we wish to model-check a formula φ of the form $\psi_1 \Rrightarrow \psi_2 \in \overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$, as defined in (13.1). Denote, again, each truth value in $\mathbb{B}_5$ as:

$$\mathbb{B}_5 = \{0000, 0001, 0011, 0111, 1111\} = \{\mathbb{B}_5[0], \mathbb{B}_5[1], \mathbb{B}_5[2], \mathbb{B}_5[3], \mathbb{B}_5[4]\}\,.$$

Let $b(L(\mathcal{G}), \varphi) \in \mathbb{B}_5$ be the computed truth value. Algorithm 1 first model-checks the corresponding LTL formula φ_4, which is of the form $\text{ltl}(4, \psi_1) \Rightarrow \text{ltl}(4, \psi_2)$. From Lemma 13.4.1 and Remark 12, this first model-checking step makes use of an automaton with the number of states as in (13.2). There are two possible outcomes:

1. the formula is violated, i.e., $b(L(\mathcal{G}), \varphi) = \mathbb{B}_5[0]$, Algorithm 1 terminates and Theorem 13.1.1 holds;

2. the formula is satisfied, i.e., $b(L(\mathcal{G}), \varphi) \succeq \mathbb{B}_5[1]$, and we need to check bit 3.

However, having checked bit 4, and given that neither ψ_1, nor ψ_2 contain a $\Rrightarrow$ operator since they belong in $\mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$, it follows from the semantics in Table 12.1 that $V_3(\sigma, \varphi) = W(\sigma, \mathrm{ltl}(3, \psi_1) \Rightarrow \mathrm{ltl}(3, \psi_2))$. We can, hence, perform a second model-checking step, using an automaton with size, again, as in (13.2), and decide if $b(L(\mathcal{G}), \varphi) = \mathbb{B}_5[1]$ or $b(L(\mathcal{G}), \varphi) \succeq \mathbb{B}_5[2]$. The two other bits are computed similarly if needed.

Overall, the number of model-checking steps that the algorithm goes through depends on $b(L(\mathcal{G}), \varphi)$. If we have exactly $\ell < 4$ bits set to 1 in the valuation, then we need $\ell + 1$ model-checking steps, i.e., to check that the bit ℓ is valued 1 and that the next bit is valued 0. The fourth verification disambiguates between the values $\mathbb{B}_5[3]$ and $\mathbb{B}_5[4]$. Hence, if $b(L(\mathcal{G}), \varphi) = \mathbb{B}_5[\ell]$, we do $\min(\ell + 1, 4)$ model-checking steps. Consequently, by using Algorithm 1, the rLTL model-checking problem for any $\varphi \in \overline{\mathrm{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ is solved by performing at most 4 LTL model-checking steps, each using an automaton with at most $\mathcal{O}\left(2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}\right)$ states. This concludes the proof of Theorem 13.1.1 and this section.

CHAPTER 14

Case studies

In the introduction we argued that system correctness is not sufficient for a good design as the system needs to also be robust. Towards this goal, we provide a series of case studies that exemplify the usefulness of rLTL when compared to standard LTL. We compare rLTL and LTL verification in terms of the ability to guarantee robustness, the information provided via verification, and the computational costs. Our case studies[1] show that model-checking in the proposed fragment $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$:

1. identifies a non-robust system, which cannot be directly done with standard LTL;
2. provides access to fine-grained information about the degree of specification violations, which can be useful towards improving the design;
3. incurs a relatively small computational overhead with respect to LTL model-checking;
4. scales similarly to the LTL model-checking with respect to the size of the given formula, although slightly more expensive.

14.1 Evrostos: the rLTL verifier

To support our theoretical contributions, i.e., identifying the fragment $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ and proving the refined complexity bounds for rLTL verification, we developed *Evrostos*[2] [ANP19].

[1] Our case studies are available at: https://github.com/janis10/evrostos/tree/master/case_studies/.

[2] Evrostos is available at: https://github.com/janis10/evrostos.

Evrostos is an open-source tool dedicated to rLTL model-checking within the fragment $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$, and it is built on top of an LTL model-checker.

On a high level, it consists of two different components. The first component is an rLTL-to-LTL translator that takes a formula $\varphi \in \overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ and returns the four corresponding LTL formulae $\varphi_j \in \text{LTL}(\mathcal{P})$, $j = 1, 2, 3, 4$. The second component implements Algorithm 1 that model-checks the LTL formulae corresponding to each bit of the truth value of an rLTL formula, and by exploiting the order of the rLTL truth values, it benefits from an early stopping criterion if the value of a bit is zero. Each model-checking step is carried out by state-of-the-art LTL model-checkers. At the time of writing this manuscript, Evrostos supports two modes:

1. `smv-mode`, which uses NuSMV[3] [CCG02] as the underlying model-checker for system models given in `smv` format; and

2. `pml-mode`, which uses SPIN[4] [Hol97] for system models given in `pml` format.

The syntax of rLTL closely resembles that of LTL to ease adoption, and in a similar manner the input format of Evrostos for rLTL formulae closely follows the one contemporary model-checkers use for LTL. The notation for the rLTL and the LTL operators used by Evrostos is as in Table 14.1.

14.2 Case study 1: Aircraft Wheel Brake System (WBS)

For our first case study, we revisit the aircraft WBS described in the Aerospace Information Report (AIR) 6110 [Aut11]. As aerospace systems have become more complex with the passing of time, it is essential that their development proceeds in a systematic way that

[3]NuSMV is available at: http://nusmv.fbk.eu.

[4]SPIN is available at: http://spinroot.com.

minimizes errors. Towards this, the Federal Aviation Administration (FAA) specifies methods and guidelines [ARP96, Aut10] for manufacturers, e.g., Boeing and Airbus, to guarantee that the development of products meets the necessary performance and safety requirements. AIR6110 provides an application of the specified processes [ARP96, Aut10] to the example of a WBS. The WBS comprises a complex hydraulic plant managing two landing gears, each with four wheels, and controlled by an independent computer system.

An extended formal verification of the WBS is found in [BCF15] and the `.smv` models used can be found in the references therein. The formal modeling and analysis are based on the integration of the contract-based design tool OCRA [CDT13], the model-checker nuXmv [CCD14], and the xSAP platform for model-based safety analysis [BBC16].

Table 14.1: Operators of rLTL and LTL in Evrostos.

rLTL			LTL		
Operator		Evrostos Symbol	Operator		Evrostos Symbol
Negation	$\neg$	!	Negation	$\neg$	!
Disjunction	$\vee$	|	Disjunction	$\vee$	|
Conjunction	$\wedge$	&	Conjunction	$\wedge$	&
Robust Implication	$\Rrightarrow$	=>	Implication	$\Rightarrow$	->
Next	$\odot$	rX	Next	$\bigcirc$	X
Robust Always	$\boxdot$	rG	Always	$\Box$	G
Robust Eventually	$\diamondsuit$	rF	Eventually	$\Diamond$	F
Robust Until	$\mathcal{U}$	rU	Until	$\mathcal{U}$	U
Robust Release	$\mathcal{R}$	rR	Release	$\mathcal{R}$	R

14.2.1 Formal specifications:

A number of safety requirements from the AIR6110 document are formalized in LTL and are expressed as reactive specifications, where the assumption is on the environment of the WBS and the guarantee is the exact safety specification [BCF15]. Here we focus on the requirement *"S18-WBS-R-0325-wheelX: never inadvertent braking of wheel X without locking"*. The environment assumption for the safety specifications is that *"at all times the power of the system is on, the power of the hydraulic pumps is on, and the hydraulic supplies maintain their nominal values"*. The above assumption and guarantee are formalized as a reactive LTL specification:

$$\Box \left(\bigwedge_{i=1}^{2} \texttt{power}_\texttt{i} \bigwedge_{i=1}^{2} \texttt{pump\_power}_\texttt{i} \bigwedge_{i=1}^{2} \texttt{hydraulic\_supply}_\texttt{i} = 10 \right) \Rightarrow \tag{14.1}$$

$$\Box\neg \left(\begin{array}{l} \neg \texttt{mechanical\_pedal}_\texttt{L} \wedge \texttt{wheel\_status} = \texttt{rolling} \\ \quad \wedge\ \texttt{wheel\_braking\_force} > 0 \wedge \texttt{ground\_speed} > 0 \end{array} \right),$$

where $\texttt{power}_\texttt{i}$, $\texttt{pump\_power}_\texttt{i}$, $\texttt{hydraulic\_supply}_\texttt{i}$ are the i-th system's power, pump power (both boolean), and hydraulic supply (integer) respectively, $\texttt{mechanical\_pedal}_\texttt{L}$ (boolean) is true if the left pedal is pressed, `ground_speed` (integer) is the aircraft's current speed relative to the ground, `wheel_status` is either rolling or stopped, and `wheel_braking_force` (integer) is the force applied by the brakes to the wheel.

The development of the WBS in AIR6110 followed four evolutionary architectures:

1. Arch1: comprises one Braking System Control Unit (BSCU) and one Hydraulic Circuit (HC) backed by an accumulator.

2. Arch2: includes additional backup components: two BSCUs, a green and a blue HC.

3. Arch3: the two BSCUs of the control system are replaced by one dual channel BSCU.

4. Arch4: accumulator placement is modified, a link from the control system validity to the selector valve in the physical system is added.

For more details see [BCF15].

14.2.2 Example scenario:

We consider the Arch4 architecture and investigate how rLTL identifies a non-robust system, which cannot be directly done in LTL. While the nominal Arch4 model is correct and robust, we introduce a modification that makes it non-robust. In particular, we inject a bug in the sensor of the left pedal that makes it periodically miss the pressing of the pedal. This results in the BSCU not always receiving an electrical signal when the left pedal is pressed. To demonstrate our case, we consider a scenario where the environment assumption is violated finitely many times. This is reasonable as during the course of a flight, there can be perturbations to the environment assumption, but we expect the assumption to stabilize and not fail catastrophically. For example, the power input of the system might be interrupted, but eventually becomes stable.

Model-checking separately the guarantee in (14.1) under the model with the sensor bug returns *false*. Model-checking separately the assumption in (14.1) under the environment scenario above returns *false*. However, model-checking the LTL specification in (14.1) under the discussed scenario returns *true*. This is a consequence of the fact that in LTL, violation of the assumption leads to vacuous satisfaction of the specification.

In contrast to LTL model-checking, we evaluate the corresponding rLTL reactive specification:

$$\boxdot\left(\bigwedge_{i=1}^{2}\texttt{power}_{\texttt{i}}\bigwedge_{i=1}^{2}\texttt{pump\_power}_{\texttt{i}}\bigwedge_{i=1}^{2}\texttt{hydraulic\_supply}_{\texttt{i}}=10\right)\Rrightarrow \tag{14.2}$$
$$\boxdot\neg\left(\begin{array}{l}\neg\texttt{mechanical\_pedal}_{\texttt{L}}\wedge\texttt{wheel\_status}=\texttt{rolling}\\ \quad\wedge\texttt{wheel\_braking\_force}>0\wedge\texttt{ground\_speed}>0\end{array}\right).$$

Using Evrostos, the resulting rLTL truth value is 0011. This is interpreted as the guarantee being both satisfied and violated infinitely often under the environment of this scenario. To be more precise, the assumption is eventually always satisfied, meaning that its truth value

is 0111. However, the sensor pedal does not always pick up the pressing of the left pedal, meaning that it misses infinitely often during a system execution and, hence, the truth value of the guarantee is 0011. By the semantics of robust implication in (11.13) we expect the truth value of the specification to be 0011, which is what Evrostos returns.

The above case study demonstrates the fact that the LTL implication cannot provide any actual information about the guarantee of a reactive specification whenever the assumption fails. Contrary, the rLTL implication does really verify whether a guarantee is satisfied, is violated, and to what degree.

14.3 Case study 2: Meaningful reactivity rLTL patterns

For the second case study, we exhibit the practicality of the proposed fragment $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ in Definition 19, and illustrate how rLTL provides more insight when a specification is violated compared to LTL. We first consider reactivity patterns of practical importance that occur commonly in the specification of concurrent and reactive systems [DAC99]. Typical behaviors include the occurrence of a given event during system execution, such as absence, existence, and universality, or the relative order in which multiple events occur during system execution, such as precedence and response. The following corollary stems from studying all the relevant LTL patterns [DAC99].

Corollary 14.3.1. *The relevant reactivity patterns [DAC99] fall under the* $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ *fragment, as described in Definition 19, when written in rLTL.*

It is the case for all these patters that the antecedent of any nested implication does not contain a $\Box$ or a $\mathcal{R}$ operator. Hence, the same holds for their rLTL counterparts. By Proposition 13.2.2, we can equivalently write any robust implication formula $\varphi \Rightarrow \psi$ in these patterns, as $\neg\varphi \vee \psi$, which makes them immediately part of the efficient fragment $\text{rLTL}_{\setminus\{\Rightarrow\}}(\mathcal{P})$. We verified this for the 97 LTL formulae in [DAC99].

Remark 14. *A number of the patterns [DAC99] are expressed using the "weak until" operator $\mathcal{W}$, which is related to the $\mathcal{U}$ operator by the LTL semantic equivalence between $p\mathcal{W}q$ and $p\ \mathcal{U}\ (q \vee \Box p)$, and to the $\mathcal{R}$ operator by the LTL semantic equivalence between $p\mathcal{W}q$ and $q\ \mathcal{R}\ (q \vee p)$. It can be verified that in the context of rLTL, only the second equivalence captures the desired meaning of a robust version of the $\mathcal{W}$ operator. Therefore, to obtain the rLTL versions of these patterns, we first replace every subformula of the form $p\mathcal{W}q$ with $q\ \mathcal{R}\ (q \vee p)$.*

14.3.1 Benchmark: Rigorous Examination of Reactive Systems (RERS)

The second goal of this case study is to showcase how rLTL provides more fine-grained information when a specification is violated, as opposed to LTL, by mapping an LTL *false* boolean value to different shades of false. To show this, we utilize the benchmarks found in the RERS Challenge [RER20]. The RERS Challenge contains a rich repository of problems of increasing complexity.

Using available benchmarks, we analyze meaningful reactive specifications that fall under the patterns [DAC99], that is 160 formulae spanning from RERS 2016 to RERS 2019. We use the provided `promela` models from the RERS LTL parallel track and evaluate the rLTL counterparts of the specifications therein using Evrostos [ANP19]. Our findings are summarized in Table 14.2.

The considered set is balanced between falsified and satisfied specifications. Focusing on the falsified formulae, one appreciates the variation of the different shades of false that rLTL provides. More specifically, Table 14.2 indicates that, empirically, it is rarely the case that a falsified reactive specification fails catastrophically. Instead, a weaker version holds most of the times. In particular, when looking only at the formulae that do admit a weaker version according to Proposition 13.2.1, the value 0000 is not observed at all. To better interpret the above analysis, consider as an example the truth value 0111. This value can be actually understood as "the safety specification holds with a delay", i.e., it is violated only finite times

Table 14.2: Occurrence frequency for each different rLTL truth value for 160 properties from the RERS benchmark.

	rLTL Truth Value				
	1111	0111	0011	0001	0000
Frequency (160 formulae)	53.75%	13.75%	25.625%	4.375%	2.5%
Frequency (70 falsified formulae that admit a weakened version)	–	31.43%	58.57%	10%	0%

over any infinite trace of the system. This information can guide the designer towards more efficiently fixing the faulty model, so as to trace the root of the problem, or can provide insight about modifying a possibly inaccurate specification.

14.4 Case study 3: Studying the complexity blowup between LTL and rLTL

We now aim to meaningfully compare the runtimes between LTL model-checking, and rLTL model-checking in the fragment $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$. Towards this, we study the *time complexity blowup*, ζ, between LTL and rLTL, which is defined below.

The complexity of the rLTL model-checking problem for any $\varphi \in \overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$, with respect to the GBA constructed for φ, is between $\mathcal{O}\left(2^{|\varphi|}\right)$ and $\mathcal{O}\left(3^{|\varphi|}\right)$ as Theorem 13.1.1 establishes. Similarly, the complexity of the LTL model-checking problem for the corresponding LTL formula φ_1 is $\mathcal{O}\left(2^{|\varphi_1|}\right)$. Recall φ_1 is obtained as the LTL version of φ simply by substituting the rLTL operators with their LTL counterparts. Let the times required to solve the LTL and the rLTL model-checking problems be t_{LTL} and t_{rLTL} respectively. We know that t_{LTL} is proportional to $2^{|\varphi_1|}$, and notice that $|\varphi| = |\varphi_1|$. Furthermore, $t_{rLTL} \geq t_{LTL}$, and, hence, we can write $t_{rLTL} = 2^{\zeta|\varphi|}$, $\zeta \geq 1$. Then, we ask what is the exponent ζ that

rLTL Times (sec.)		
$\min(t_{\text{rLTL}})$	$\max(t_{\text{rLTL}})$	$\text{mean}(t_{\text{rLTL}})$
0.26	726.8	66.3
LTL Times (sec.)		
$\min(t_{\text{LTL}})$	$\max(t_{\text{LTL}})$	$\text{mean}(t_{\text{LTL}})$
0.04	291.0	26.3
Time Complexity Blowup		
$\min(\zeta)$	$\max(\zeta)$	$\text{mean}(\zeta)$
1.016	1.122	1.058

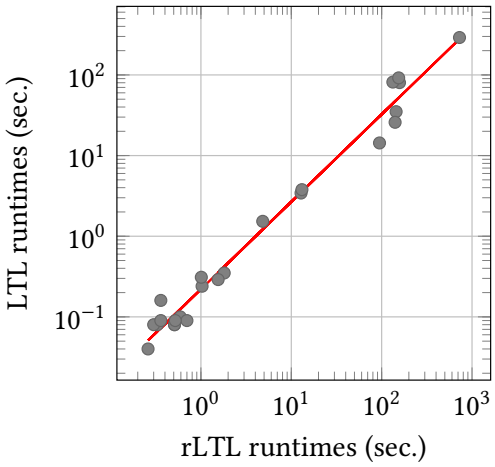


Figure 14.1: Automated Air Traffic Control System. Left: minimum, maximum, and mean runtimes for rLTL and LTL model-checking, and time complexity blowup between rLTL and LTL. Right: Comparison between runtimes for rLTL and LTL (logarithmic scale). Computed over 24 experiments.

describes the overhead, i.e., what is the time complexity blowup. From the expressions above we obtain:

$$\zeta = 1 + \frac{\log_2\left(\frac{t_{rLTL}}{t_{LTL}}\right)}{|\varphi|}.$$

Since the time complexity of rLTL model-checking for the proposed fragment $\overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$ is proportional to at most $3^{|\varphi|}$, we have an upper bound for ζ of $\log_2(3) = 1.58$.

We use as a benchmartk the model of an Automated Air Traffic Control System [ZR14], designed for the Automated Airspace Concept (AAC), and the specifications therein. ACC is a high-level generic framework proposed as a candidate for the Next Generation Air Traffic Control System, which was under development at NASA. The goal of ACC is to always ensure the safe separation of commercial aircrafts within a given airspace sector, in order to prevent potential collisions. Figure 14.1 shows on the left the minimum, maximum, and

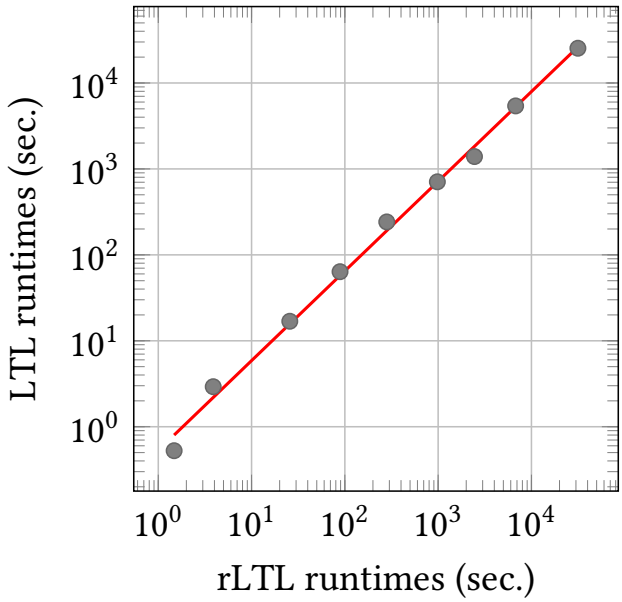


Figure 14.2: Scalability comparison: model-checking runtimes for (14.3) and (14.4) for different number of philosophers $n = 1, \ldots, 9$ (logarithmic scale).

mean runtimes to model-check 12 safety and reactivity specifications, over 2 system models (original and abstract model) for a total of 24 experiments, and on the right how these runtimes are distributed. At a first glance, rLTL verification is somewhat more expensive computationally, which is expected as its runtime is proportional to at most $3^{|\varphi|}$ for a formula $\varphi \in \overline{\text{rLTL}}_{\setminus\{\Rightarrow\}}(\mathcal{P})$. However, observe that the time complexity blowup, ζ, is well below this upper bound, even at its maximum value in this benchmark, meaning that the empirical time complexity of rLTL for the fragment we consider is close to that of LTL.

14.5 Case study 4: Scalability

For our last set of experiments, we present a scalability case study for rLTL within the proposed fragment. To this end, we select the well-known model of the *dining philosophers* and consider the following specification, ϕ, saying that if whenever a philosopher is ready

they eventually eat, then the first philosopher will eventually eat:

$$\left(\bigwedge_{i=1}^{n} \Box\Diamond ready_i \Rightarrow \Box\Diamond eat_i\right) \Rightarrow eat_0.$$

The above LTL formula contains nested implications with an $\Box$ operator, which are not allowed in the proposed fragment $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$. However, we can equivalently rewrite the LTL formula as:

$$\left(\bigwedge_{i=1}^{n} \neg(\Box\Diamond ready_i) \vee \Box\Diamond eat_i\right) \Rightarrow eat_0, \tag{14.3}$$

and then the corresponding rLTL specification φ is:

$$\left(\bigwedge_{i=1}^{n} \neg(\boxdot \diamondsuit ready_i) \vee \boxdot \diamondsuit eat_i\right) \Rrightarrow eat_0. \tag{14.4}$$

This specification presents a complex temporal structure as it contains multiple $\boxdot$ and $\diamondsuit$ operators, as well as an $\Rrightarrow$ operator on the outermost level and, hence, is an appropriate candidate for our case study. We fix the model to have ten philosophers and then record the runtimes for evaluating (14.3) and (14.4) for $n = 1, \ldots, 9$. Our findings are summarized in Fig. 14.2. The specification φ belongs to $\overline{\text{rLTL}}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ and by Theorem 13.1.1 can be model-checked using automata of size $\mathcal{O}\left(2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}\right)$. We can appreciate that the runtimes for model-checking the rLTL specification scale in the same manner as those for the corresponding LTL model-checking, although slightly higher as expected by the aforementioned automaton size. This validates our theoretical results. This case study is also in agreement with the results presented in Fig. 14.1 when studying the complexity blow-up between LTL and rLTL.

CHAPTER 15

Conclusion

The logic rLTL provides a means to formally reason about both correctness and robustness in system design. While its syntax closely resembles that of LTL to ease its adoption, its semantics embeds a notion of robustness expressing that small deviations from the assumptions made at design time should lead to, at most, small violations of the design specifications. In this paper we presented a large fragment of rLTL, for which the verification problem can be efficiently solved by using an automaton of size $\mathcal{O}\left(2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}\right)$, where $\kappa(\varphi)$ measures the number of unique subformulae of φ that contain always and release operators. This bound is closer to the LTL bound of $\mathcal{O}\left(2^{|\varphi|}\right)$ and an improvement to the previously known bound of $\mathcal{O}\left(5^{|\varphi|}\right)$. Moreover, at the time of publication there is no known non-trivial lower bound and finding such bound is an open problem. The usefulness of this fragment has been demonstrated by a number of case studies showing its expressiveness, the ability to capture robustness, and the benefits of the information the designer gains from the 5-valued semantics towards refining the system and/or the specifications. Moreover, these advantages come at low computational overhead with respect to LTL model-checking, and a small learning curve from the designer as the syntax of rLTL closely mirrors that of LTL.

APPENDIX A

Appendix

A.1 Proof of Lemma 12.1.1

Proof. By using the ltl operator defined in Table 12.1 one translates any rLTL formula $\varphi \in \mathrm{rLTL}(\mathcal{P})$ to four LTL formualae $\mathrm{ltl}(j,\varphi) \in \mathrm{LTL}(\mathcal{P})$, $j = 1,2,3,4$, as defined in (12.3). We show that the number of the unique subformulae resulting by the translation is linear to the size of φ, i.e.,:

$$\mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big)\right) \leq c|\varphi|, \tag{A.1}$$

for some $c \in \mathbb{N}$. We use an induction argument and show that (A.1) holds for $c = 12$.

Base Case. Take any $\varphi \in \mathrm{rLTL}(\mathcal{P})$ of length 1, i.e., φ is $p \in \mathcal{P}$. Then the claim holds straightforwardly as: $\mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}(\mathrm{ltl}(j,p))\right) = \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}(p)\right) = |p| \leq 12|p|$.

Induction. First, consider any $\varphi \in \mathrm{rLTL}(\mathcal{P})$ of the form $\psi_1 \ \mathcal{R} \ \psi_2$ and note that

$|\varphi| = |\psi_1| + |\psi_2| + 1$. Then:

$$\begin{aligned}
\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big) &= \bigcup_{j=1}^{4} \mathrm{cl}(\mathrm{ltl}(j, \psi_1 \mathcal{R} \psi_2)) \\
&= \mathrm{cl}\big(\mathrm{ltl}(1,\psi_1) \mathcal{R} \mathrm{ltl}(1,\psi_2)\big) \cup \mathrm{cl}\big(\Diamond \mathrm{ltl}(2,\psi_2) \vee \Diamond\Box \mathrm{ltl}(2,\psi_1)\big) \\
&\qquad \cup \mathrm{cl}\big(\Diamond \mathrm{ltl}(3,\psi_2) \vee \Box\Diamond \mathrm{ltl}(3,\psi_1)\big) \cup \mathrm{cl}\big(\Diamond \mathrm{ltl}(4,\psi_2) \vee \Diamond \mathrm{ltl}(4,\psi_1)\big) \\
&= \bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_1)\big) \bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_2)\big) \cup \big\{\mathrm{ltl}(1,\psi_1) \mathcal{R} \mathrm{ltl}(1,\psi_2)\big\} \\
&\qquad \cup \big\{\Diamond \mathrm{ltl}(2,\psi_2), \Box \mathrm{ltl}(2,\psi_1), \Diamond\Box \mathrm{ltl}(2,\psi_1), \Diamond \mathrm{ltl}(2,\psi_2) \vee \Diamond\Box \mathrm{ltl}(2,\psi_1)\big\} \\
&\qquad \cup \big\{\Diamond \mathrm{ltl}(3,\psi_2), \Diamond \mathrm{ltl}(3,\psi_1), \Box\Diamond \mathrm{ltl}(3,\psi_1), \Diamond \mathrm{ltl}(3,\psi_2) \vee \Box\Diamond \mathrm{ltl}(3,\psi_1)\big\} \\
&\qquad \cup \big\{\Diamond \mathrm{ltl}(4,\psi_2), \Diamond \mathrm{ltl}(4,\psi_1), \Diamond \mathrm{ltl}(4,\psi_2) \vee \Diamond \mathrm{ltl}(4,\psi_1)\big\}.
\end{aligned}$$

In turn the above implies that:

$$\begin{aligned}
\mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big)\right) &\leq \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_1)\big)\right) + \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_2)\big)\right) + 12 \\
&\leq 12|\psi_1| + 12|\psi_2| + 12 = 12(|\psi_1| + |\psi_2| + 1) = 12|\varphi|,
\end{aligned}$$

where in the last inequality we used the induction hypothesis from (A.1) for $c = 12$. The other operators, with the exception of implication and negation, are similar and, thus, omitted for the sake of conciseness.

We next consider $\varphi \in \mathrm{rLTL}(\mathcal{P})$ of the form $\psi_1 \Rrightarrow \psi_2$ and observe, given Table 12.1, that:

$$\mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big) \supseteq \mathrm{cl}(\mathrm{ltl}(i,\varphi)),\ i \geq j,\ i,j \in \{1,2,3,4\}.$$

Then we have that:

$$\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big) = \mathrm{cl}\big(\mathrm{ltl}(1,\varphi)\big) = \mathrm{cl}(\mathrm{ltl}(1,\psi_1 \Rrightarrow \psi_2)) = \mathrm{cl}\left(\bigwedge_{j=1}^{4} \mathrm{ltl}(j,\psi_1) \Rightarrow \mathrm{ltl}(j,\psi_2)\right)$$

$$= \bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_1)\big) \bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_2)\big)$$

$$\bigcup_{j=1}^{4} \big\{\mathrm{ltl}(j,\psi_1) \Rightarrow \mathrm{ltl}(j,\psi_2)\big\} \bigcup_{k=1}^{3} \left\{\bigwedge_{j=k}^{4} \mathrm{ltl}(j,\psi_1) \Rightarrow \mathrm{ltl}(j,\psi_2)\right\}$$

$$\Rightarrow \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big)\right) \leq \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_1)\big)\right) + \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\psi_2)\big)\right) + 7$$

$$\leq 12|\psi_1| + 12|\psi_2| + 7 = 12(|\psi_1| + |\psi_2| + 1) = 12|\varphi|,$$

The case of negation is similar as $\mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big) = \mathrm{cl}(\mathrm{ltl}(i,\varphi)),\ i \neq j,\ i,j \in \{1,2,3,4\}$. This concludes the proof by induction.

Finally, $|\mathrm{ltl}(j,\varphi)| \leq \mathrm{card}\left(\bigcup_{j=1}^{4} \mathrm{cl}\big(\mathrm{ltl}(j,\varphi)\big)\right) \leq 12|\varphi|$, $j \in \{1,2,3,4\}$, which proves that the translation complexity is linear in the size of the formula, i.e., linear in the number of its unique subformulae, and concludes the proof. □

A.2 Proof of Lemma 13.4.1

Proof. The proof follows from the rLTL semantics as defined in Table 12.1, Proposition 13.3.1, and Corollary 13.3.2. We proceed by induction for all operators except for the $\mathcal{R}$ operator, for which we construct a specialized tester.

Base Case. Take a formula $\varphi \in \mathrm{rLTL}_{\setminus\{\Rrightarrow\}}(\mathcal{P})$ of length 1, i.e., φ is $p \in \mathcal{P}$. Then we get $|\mathrm{T}_p| = 2^{|\varphi|} = 2$, which is the higher possible number of states here, and the claim holds.

Induction. First, consider formulae $\varphi \in \mathrm{rLTL}(\mathcal{P})$ of the form $\mathrm{op}(\psi)$, where op is any unary rLTL operator. Note that $|\varphi| = |\psi| + 1$.

1. If op is $\boxdot$, from (13.13) we obtain for $j \in \{1,2,3,4\}$:

$$\big|\mathrm{T}_{\mathrm{ltl}(j,\boxdot\psi)}\big| \leq 3\,\big|\mathrm{T}_{\mathrm{ltl}(j,\psi)}\big| \leq 3 \cdot 2^{|\psi|-\kappa(\psi)} 3^{\kappa(\psi)} \leq 2^{|\psi|-\kappa(\psi)+1-1} 3^{\kappa(\psi)+1} = 2^{|\varphi|-\kappa(\varphi)} 3^{\kappa(\varphi)},$$

which holds for the second and third bit.

2. If op is any other unary rLTL operator, from (13.11), (13.12) we obtain for $j \in \{1,2,3,4\}$:

$$\left|\mathrm{T}_{\mathrm{ltl}(j,\mathrm{op}(\psi))}\right| \leq 2\left|\mathrm{T}_{\mathrm{ltl}(i,\psi)}\right| \leq 2^{|\psi|+1-\kappa(\psi)}3^{\kappa(\psi)} = 2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)},$$

where $i = 1$ if op is $\neg$, and otherwise $i = j$.

Consider now $\varphi \in \mathrm{rLTL}(\mathcal{P})$ of the form $\mathrm{op}(\psi_1, \psi_2)$, where op is a binary operator. Its length is $|\varphi| = |\psi_1| + |\psi_2| + 1$.

1. If op is either $\wedge$ or $\vee$, from (13.14) we obtain for $j \in \{1,2,3,4\}$:

$$\left|\mathrm{T}_{\mathrm{ltl}(j,\mathrm{op}(\psi_1,\psi_2))}\right| \leq \left|\mathrm{T}_{\mathrm{ltl}(j,\psi_1)}\right| \cdot \left|\mathrm{T}_{\mathrm{ltl}(j,\psi_2)}\right| \leq 2^{|\psi_1|+|\psi_2|-\kappa(\psi_1)-\kappa(\psi_2)}3^{\kappa(\psi_1)+\kappa(\psi_2)} \leq 2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}.$$

2. If op is $\mathcal{U}$, from (13.15) we obtain for $j \in \{1,2,3,4\}$:

$$\left|\mathrm{T}_{\mathrm{ltl}(i,\psi_1\mathcal{U}\psi_2)}\right| \leq 2\left|\mathrm{T}_{\mathrm{ltl}(j,\psi_1)}\right| \cdot \left|\mathrm{T}_{\mathrm{ltl}(j,\psi_2)}\right| \leq 2^{|\psi_1|+|\psi_2|+1-\kappa(\psi_1)-\kappa(\psi_2)}3^{\kappa(\psi_1)+\kappa(\psi_2)} \leq 2^{|\varphi|-\kappa(\varphi)}3^{\kappa(\varphi)}.$$

This concludes the proof by induction.

Finally, we need to prove the same bounds for the $\mathcal{R}$ operator. This case is slightly more tricky, but it suffices to show that for any $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$, there exist appropriate testers such that:

$$|\mathrm{T}_{\varphi\mathcal{R}\psi}| \leq 3 \cdot |\mathrm{T}_\varphi| \cdot |\mathrm{T}_\psi|, \tag{A.2}$$

$$|\mathrm{T}_{\Diamond\Box\psi\vee\Diamond\varphi}| \leq 3 \cdot |\mathrm{T}_\varphi| \cdot |\mathrm{T}_\psi|, \tag{A.3}$$

$$|\mathrm{T}_{\Box\Diamond\psi\vee\Diamond\varphi}| \leq 3 \cdot |\mathrm{T}_\varphi| \cdot |\mathrm{T}_\psi|, \tag{A.4}$$

$$|\mathrm{T}_{\Diamond\psi\vee\Diamond\varphi}| \leq 3 \cdot |\mathrm{T}_\varphi| \cdot |\mathrm{T}_\psi|, \tag{A.5}$$

which by Table 12.1 are the testers for the corresponding 4 LTL formulae due to the $\mathcal{R}$ operator.

1. Proof of (A.2): The inequality follows from (13.15). Notice that we can replace the constant 3 by a 2 here.

2. Proof of (A.3): The LTL formulae $\Diamond\Box q \vee \Diamond p$ and $\Diamond(\Box q \vee p)$ are semantically equivalent for any $p, q \in \mathcal{P}$. Thus, we construct a tester for $\Diamond(\Box q \vee p)$ in Figure **??** by following

the composition rules. This tester has at most 3 nodes that assign the same value to the atomic propositions p and q: two states assign $(x_p, x_q, x_\varphi) = (0, 1, 1)$, and one state assigns $(x_p, x_q, x_\varphi) = (0, 1, 0)$. From this and Proposition 13.3.1 we conclude (A.3).

3. Proof of (A.4): We provide a *specialized tester* for any φ of the form $\square\diamond q \vee \diamond p$ in Figure **??** and prove its correctness, i.e., show that the tester is both *sound* and *complete* [PZ08, Section 5]. Recall from Definition 21 that given a computation γ, we let $\sigma(\gamma) \in \left(2^{\mathcal{P}}\right)^\omega$ be the word $\sigma(\gamma) = \sigma_0(\gamma)\sigma_1(\gamma)\ldots$ where $\sigma_t(\gamma)$ is the subset of $\mathcal{P}$ defined by $p \in \sigma_t(\gamma)$, if and only if, $x_p^{(t)} = 1$. Soundness is defined as follows.

Definition 24 (Soundness)**.** *Given a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, a tester T_φ is* sound *if for all computations $\gamma = x^{(0)}x^{(1)}\ldots$ of T_φ, we have that $x_\varphi^{(t)} = 1$, if and only if, $W(\sigma(\gamma)_{t\ldots}, \varphi) = 1$, where $\sigma(\gamma)_{t\ldots}$ is the suffix of $\sigma(\gamma)$ starting at the t-th position.*

To prove soundness we consider all possible initial states for computations $\gamma = x^{(0)}x^{(1)}\ldots$ of T_φ:

- If $x^{(0)}$ is any of the three states on the left, then at time $t = 0$, we have $W(\sigma(\gamma), \varphi) = 0$.
- If $x^{(0)}$ is any of the two middle states, then $W(\sigma(\gamma), p) = 1$, hence $W(\sigma(\gamma), \varphi) = 1$.
- If $x^{(0)}$ is any of the rightmost states, then either the computations visit one of the middle states, or never do so. In the first case, we have $W(\sigma(\gamma), \diamond p) = 1$. In the second case, they had to visit the bottom right node infinitely often due to the justice requirements. Therefore, $W(\sigma(\gamma), \square\diamond q) = 1$. Combining the two cases results in $W(\sigma(\gamma), \varphi) = 1$.

We conclude that the tester is sound and move onto proving its completeness.

Definition 25 (Completeness)**.** *Given a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, a tester T_φ is* complete *if for any word $\sigma \in \left(2^{\mathcal{P}}\right)^\omega$, there exists a computation γ of T, such that $\forall t \geq 0$, $x_\varphi^{(t)} = 1$, if and only if, $W(\sigma_{t\ldots}, \varphi) = 1$, where $\sigma_{t\ldots}$ is the suffix of σ starting at the t-th position.*

To prove completeness, we consider words $\sigma \in \{p, q\}^\omega$ and different cases based on which subformulae of φ they satisfy:

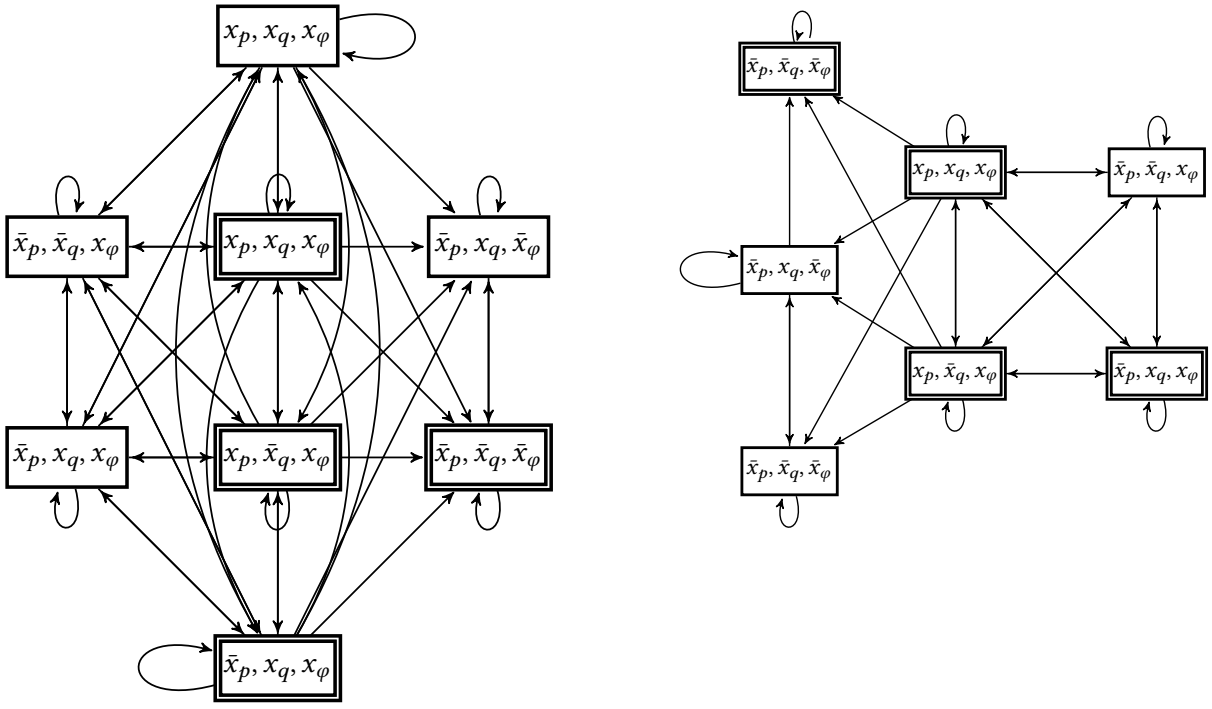
- If σ satisfies $\Diamond p$, pick a computation with initial state either in the middle or in the right parts of the tester. The computation remains in these parts and visits middle states as many times as p occurs in σ. If it occurs infinitely often, then the computation satisfies the justice requirements. Otherwise, there is a $t > 0$ after which $\sigma_{t\ldots}$ satisfies $\Box\neg p$, which is discussed next.

- If σ satisfies $\Box\neg p$, we are only interested in the evolution of the x_q variables. In particular, the corresponding computations are to remain either in the left or in the right parts of the tester. Hence, we can ignore the two middle nodes here. To conclude, observe that by doing so, we obtain a tester for $\Box\Diamond q$ (see Figure 13.1d and the fact that $\Diamond\Box\psi$ is equivalent to $\neg\Box\Diamond\neg\psi$).

The tester is therefore sound and complete. The constant 3 in (A.4) is obtained by counting the corresponding number of nodes in T_φ for all combinations of x_p and x_q, and then using Proposition 13.3.1. There are 3 nodes such that $x_p = x_q = 0$.

4. Proof of (A.5): The proof here is direct when considering the following LTL equalities $\Diamond p \vee \Diamond q = \Diamond(p \vee q) = \mathit{true}\, \mathcal{U}\, (p \vee q)$ and equations (13.14) and (13.15).

This proves the bound for the $\mathcal{R}$ operator and concludes the proof. □



(a) Tester $\mathrm{T}_{\Diamond\Box q \vee \Diamond p}$.

(b) Tester $\mathrm{T}_{\Box\Diamond q \vee \Diamond p}$.

Figure A.1: All states are initial, double line states are contained in the set of justice requirements.

REFERENCES

[ABK16] Shaull Almagor, Udi Boker, and Orna Kupferman. "Formally Reasoning About Quality." *J. ACM*, **63**(3), June 2016.

[AK14] Shaull Almagor and Orna Kupferman. "Latticed-LTL Synthesis in the Presence of Noisy Inputs." In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures*, pp. 226–241, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[ALO21a] Tzanis Anevlavis, Zexiang Liu, Necmiye Ozay, and Paulo Tabuada. "Controlled invariant sets: implicit closed-form representations and applications.", 2021.

[ALO21b] Tzanis Anevlavis, Zexiang Liu, Necmiye Ozay, and Paulo Tabuada. "An enhanced hierarchy for (robust) controlled invariance." In *2021 American Control Conference (ACC)*, pp. 4860–4865, 2021.

[AM06] Panos J. Antsaklis and Anthony Michel. *Linear Systems*. Birkhäuser Basel, 1st edition, 2006.

[ANP19] Tzanis Anevlavis, Daniel Neider, Matthew Philippe, and Paulo Tabuada. "Evrostos: The rLTL Verifier." In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pp. 218–223, New York, NY, USA, 2019. Association for Computing Machinery.

[APN18] T. Anevlavis, M. Philippe, D. Neider, and P. Tabuada. "Verifying rLTL formulas: now faster than ever before!" In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1556–1561, Dec 2018.

[APN22] Tzanis Anevlavis, Matthew Philippe, Daniel Neider, and Paulo Tabuada. "Being Correct Is Not Enough: Efficient Verification Using Robust Linear Temporal Logic." *ACM Trans. Comput. Logic*, **23**(2), jan 2022.

[ARP96] Society of Automotive Engineers (SAE) ARP4761. "Guidelines and methods for conducting the safety assessment process on airborne systems and equipments." *USA: The Engineering Society for Advancing Mobility Land Sea Air and Space*, 1996.

[AT19] T. Anevlavis and P. Tabuada. "Computing controlled invariant sets in two moves." In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 6248–6254, 2019.

[AT20] Tzanis Anevlavis and Paulo Tabuada. "A Simple Hierarchy for Computing Controlled Invariant Sets." In *Proceedings of the 23rd International Conference on*

Hybrid Systems: Computation and Control, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[Aut10] Society of Automotive Engineers (SAE). "Guidelines for development of civil aircraft and systems (ARP4754A)." *SAE International*, 2010.

[Aut11] Society of Automotive Engineers (SAE). "Contiguous Aircraft/System Development Process Example (AIR6110)." 2011.

[BBB19] Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments." In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1758–1765. IEEE, 2019.

[BBC16] Benjamin Bittner, Marco Bozzano, Roberto Cavada, Alessandro Cimatti, Marco Gario, Alberto Griggio, Cristian Mattarei, Andrea Micheli, and Gianni Zampedri. "The xSAP safety analysis platform." In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 533–539. Springer, 2016.

[BCE19] R. Bloem, H. Chockler, M. Ebrahimi, and O. Strichman. "Synthesizing Reactive Systems Using Robustness and Recovery Specifications." In *2019 Formal Methods in Computer Aided Design (FMCAD)*, pp. 147–151, Oct 2019.

[BCF15] M. Bozzano, A. Cimatti, A. Fernandes Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta. "Formal Design and Safety Analysis of AIR6110 Wheel Brake System." In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification*, pp. 518–535, Cham, 2015. Springer International Publishing.

[BCG10] R. Bloem, K. Chatterjee, K. Greimel, T.A. Henzinger, and B. Jobstmann. "Robustness in the Presence of Liveness." In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Computer Aided Verification*, volume 6174 of *Lecture Notes in Computer Science*, pp. 410–424. Springer Berlin Heidelberg, 2010.

[BCG14] Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. "Synthesizing robust systems." *Acta Informatica*, **51**(3):193–220, 2014.

[BCV21] Suguman Bansal, Krishnendu Chatterjee, and Moshe Y. Vardi. "On Satisficing in Quantitative Games." In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 20–37, Cham, 2021. Springer International Publishing.

[Ber72] Dimitri Bertsekas. "Infinite time reachability of state-space regions by using feedback control." *Automatic Control, IEEE Transactions on*, **AC-17**:604 – 613, 11 1972.

[BJP12] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar. "Synthesis of reactive(1) designs." *Journal of Computer and System Sciences*, **78**(3):911–938, January 2012.

[BKL08] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking*. MIT press, 2008.

[Bru70] Pavol Brunovský. "A classification of linear controllable systems." *Kybernetika*, **6**:173–188, 1970.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CCD14] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. "The nuXmv Symbolic Model Checker." In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification*, pp. 334–342, Cham, 2014. Springer International Publishing.

[CCG02] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. "NuSMV 2: An OpenSource Tool for Symbolic Model Checking." In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pp. 359–364, London, UK, UK, 2002. Springer-Verlag.

[CDT13] Alessandro Cimatti, Michele Dorigatti, and Stefano Tonetta. "OCRA: A Tool for Checking the Refinement of Temporal Contracts." In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, ASE'13, pp. 702–705. IEEE Press, 2013.

[CG86] M. Cwikel and P.-O. Gutman. "Convergence of an algorithm to find maximal state constraint sets for discrete-time linear dynamical systems with bounded controls and states." *IEEE Transactions on Automatic Control*, **31**(5):457–459, 1986.

[CGL10] Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. "Continuity Analysis of Programs." In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, pp. 57–70, New York, NY, USA, 2010. Association for Computing Machinery.

[CGP99] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[CHV18] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of Model Checking*. Springer, 2018.

[CMP93] E. Chang, Z. Manna, and A. Pnueli. "The Safety-Progress Classification." In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, volume 94 of *NATO ASI Series*, pp. 143–202. Springer Verlag, 1993.

[CON] CONIX Research Center. "Augmented Reality Edge Networking Architecture – ARENA.".

[DAC99] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. "Patterns in Property Specifications for Finite-State Verification." In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pp. 411–420, New York, NY, USA, 1999. Association for Computing Machinery.

[DC71] J Duncan Glover and Fred C. Schweppe. "Control of Linear Dynamic Systems with Set Constrained Disturbances." *Automatic Control, IEEE Transactions on*, **16**:411 – 423, 11 1971.

[DM10] Alexandre Donzé and Oded Maler. "Robust Satisfaction of Temporal Logic over Real-Valued Signals." In Krishnendu Chatterjee and Thomas A. Henzinger, editors, *Formal Modeling and Analysis of Timed Systems*, pp. 92–106, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[FP09] Georgios E. Fainekos and George J. Pappas. "Robustness of temporal logic specifications for continuous-time signals." *Theoretical Computer Science*, **410**(42):4262–4291, September 2009.

[GC87] P.-O. Gutman and M. Cwikel. "An algorithm to find maximal state constraint sets for discrete-time linear dynamical systems with bounded controls and states." *IEEE Transactions on Automatic Control*, **32**(3):251–254, 1987.

[GF19] A. Gupta and P. Falcone. "Full-Complexity Characterization of Control-Invariant Domains for Systems With Uncertain Parameter Dependence." *IEEE Control Systems Letters*, **3**(1):19–24, 2019.

[GKF19] Ankit Gupta, Hakan Köroğlu, and Paolo Falcone. "Computation of low-complexity control-invariant sets for systems with uncertain parameter dependence." *Automatica*, **101**:330–337, 2019.

[GU19] Jerzy Grzybowski and R Urbański. "Order cancellation law in the family of bounded convex sets." *Journal of Global Optimization*, pp. 1–12, 2019.

[Gur20] LLC Gurobi Optimization. "Gurobi Optimizer Reference Manual.", 2020.

[H98] P. Hájeck. *Metamathematics of Fuzzy Logic*, volume 4 of *Trends in Logic - Studia Logica Library*. Kluwer Academic Publishers, 1998.

[HKJ13] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. "Multi-Parametric Toolbox 3.0." In *Proc. of the European Control Conference*, pp. 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/~mpt`.

[Hol97] G.J. Holzmann. "The model checker Spin." *IEEE Transactions on Software Engineering*, **23**(5):279–295, 1997.

[JT51] B. Jonsson and A. Tarski. "Boolean Algebras with Operators. Part I." *American Journal of Mathematics*, **73**(4):891 – 939, 1951.

[KHJ14] Milan. Korda, Didier. Henrion, and Colin N. Jones. "Convex Computation of the Maximum Controlled Invariant Set For Polynomial Control Systems." *SIAM Journal on Control and Optimization*, **52**(5):2944–2969, 2014.

[KL07] Orna Kupferman and Yoad Lustig. "Lattice Automata." In Byron Cook and Andreas Podelski, editors, *Verification, Model Checking, and Abstract Interpretation*, pp. 199–213, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[KPR98] Yonit Kesten, Amir Pnueli, and Li-on Raviv. "Algorithmic verification of linear temporal logic specifications." In *International Colloquium on Automata, Languages, and Programming*, pp. 1–16. Springer, 1998.

[KPV09] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. "From liveness to promptness." *Formal Methods in System Design*, **34**(2):83–103, 2009.

[Lau04] Alan J. Laub. *Matrix Analysis For Scientists And Engineers*. Society for Industrial and Applied Mathematics, USA, 2004.

[LO21] Zexiang Liu and Necmiye Ozay. "Safe Online Planning in Unknown Nonconvex Environments with Implicit Controlled Invariant Sets." *IFAC-PapersOnLine*, **54**(5):163–168, 2021. 7th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2021.

[Lof04] Johan Lofberg. "YALMIP: A toolbox for modeling and optimization in MATLAB." In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pp. 284–289. IEEE, 2004.

[Lof12] J. Lofberg. "Big-M and convex hulls.", 2012.

[LP85] Orna Lichtenstein and Amir Pnueli. "Checking that finite state concurrent programs satisfy their linear specification." In *Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pp. 97–107. ACM, 1985.

[LS15] Mircea Lazar and Veaceslav Spinu. "Finite-step Terminal Ingredients for Stabilizing Model Predictive Control." *IFAC-PapersOnLine*, **48**(23):9–15, 2015. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[LTJ18] Benoît Legat, Paulo Tabuada, and Raphaël M. Jungers. "Computing controlled invariant sets for hybrid systems with applications to model-predictive control." In *6th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2018, Oxford, UK, July 11-13, 2018*, pp. 193–198, 2018.

[MD13] M. W. Mueller and R. D'Andrea. "A model predictive controller for quadrocopter state interception." In *2013 European Control Conference (ECC)*, pp. 1383–1389, July 2013.

[MHO18] Sarmad Munir, Morten Hovd, and Sorin Olaru. "Low complexity constrained control using higher degree Lyapunov functions." *Automatica*, **98**:215 – 222, 2018.

[MNS20] Corto Mascle, Daniel Neider, Maximilian Schwenger, Paulo Tabuada, Alexander Weinert, and Martin Zimmermann. "From LTL to rLTL Monitoring: Improved Monitorability through Robust Semantics." In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[MOB18] Nathan Michel, Sorin Olaru, Sylvain Bertrand, Giorgio Valmorbida, and Didier Dumur. "Invariant set design for constrained discrete-time linear systems with bounded matched disturbance." *IFAC-PapersOnLine*, **51**(25):55–60, 2018.

[MP87] Z. Manna and A. Pnueli. "A Hierarchy of Temporal Properties." In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pp. 205–205, New York, NY, USA, 1987. ACM.

[MS09] R. Majumdar and I. Saha. "Symbolic Robustness Analysis." In *2009 30th IEEE Real-Time Systems Symposium*, pp. 355–363, Dec 2009.

[MSR05] D.Q. Mayne, M.M. Seron, and S.V. Raković. "Robust model predictive control of constrained linear systems with bounded disturbances." *Automatica*, **41**(2):219–224, 2005.

[NNZ21] Satya Prakash Nayak, Daniel Neider, and Martin Zimmermann. *Adaptive Strategies for rLTL Games*. Association for Computing Machinery, New York, NY, USA, 2021.

[NPM99] V. Novák, I. Perfilieva, and J. Mŏckoř. *Mathematical Principles of Fuzzy Logic*. Kluwer Academic Publishers, 1999.

[NWZ19] Daniel Neider, Alexander Weinert, and Martin Zimmermann. "Robust, Expressive, and Quantitative Linear Temporal Logics: Pick any Two for Free." *Electronic Proceedings in Theoretical Computer Science*, **305**:1–16, Sep 2019.

[OTH19] A. Oustry, M. Tacchi, and D. Henrion. "Inner Approximations of the Maximal Positively Invariant Set for Polynomial Dynamical Systems." *IEEE Control Systems Letters*, **3**(3):733–738, July 2019.

[PAT21] Luigi Pannocchi, Tzanis Anevlavis, and Paulo Tabuada. "Trust your supervisor: quadrotor obstacle avoidance using controlled invariant sets." In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9219–9224, 2021.

[Pri09] G. Priest. "Dualising Intuitionist Logic." *Principia*, **13**(2):165 – 184, 2009.

[PW97] D. Peled and T. Wilke. "Stutter-Invariant Temporal Properties are Expressible Without the Next-Time Operator." *Inf. Process. Lett.*, **63**:243–246, 1997.

[PZ08] Amir Pnueli and Aleksandr Zaks. "On the merits of temporal testers." In *25 Years of Model Checking*, pp. 172–195. Springer, 2008.

[RBN16] Alena Rodionova, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. "Temporal Logic as Filtering." In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, pp. 11–20, New York, NY, USA, 2016. Association for Computing Machinery.

[RER20] RERS. "Rigorous Examination of Reactive Systems (RERS) Challenge." 2010-2020.

[RKK05] Sasa V Rakovic, Eric C Kerrigan, Konstantinos I Kouramas, and David Q Mayne. "Invariant approximations of the minimal robust positively invariant set." *IEEE Transactions on automatic control*, **50**(3):406–410, 2005.

[RT17] M. Rungger and P. Tabuada. "Computing Robust Controlled Invariant Sets of Linear Systems." *IEEE Transactions on Automatic Control*, **62**(7):3665–3670, July 2017.

[Sch02] Philippe Schnoebelen. "The Complexity of Temporal Logic Model Checking." *Advances in modal logic*, **4**(393-436):35, 2002.

[SG12] Mohamed Amin Ben Sassi and Antoine Girard. "Controller synthesis for robust invariance of polynomial dynamical systems using linear programming." *Systems & Control Letters*, **61**(4):506 – 512, 2012.

[THO01] John Ronald Reuel Tolkien, Ian Holm, and Stephen Oliver. *The Lord of the Rings*. HarperCollins London, 2001.

[TJ15] F. Tahir and I. M. Jaimoukha. "Low-Complexity Polytopic Invariant Sets for Linear Systems Subject to Norm-Bounded Uncertainty." *IEEE Transactions on Automatic Control*, **60**(5):1416–1421, 2015.

[TN15] Paulo Tabuada and Daniel Neider. "Robust Linear Temporal Logic." 1510.08970. arXiv, 2015.

[TN16] Paulo Tabuada and Daniel Neider. "Robust Linear Temporal Logic." In *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, 2016.

[TTC13] Y.-K. Tsay, M.-H. Tsai, J.-S. Chang, Y.-W. Chang, and C.-S. Liu. "Büchi Store: An Open Repository of ω-Automata." *International Journal on Software Tools for Technology Transfer*, **2**(15):109–123, 2013.

[Var11] Moshe Y Vardi. "The rise and fall of LTL." *GandALF*, **54**, 2011.

[VW86] Moshe Y Vardi and Pierre Wolper. "An automata-theoretic approach to automatic program verification." In *Proceedings of the First Symposium on Logic in Computer Science*, pp. 322–331. IEEE Computer Society, 1986.

[WET15] Min Wen, Rüdiger Ehlers, and Ufuk Topcu. "Correct-by-synthesis reinforcement learning with temporal logic constraints." In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4983–4990, 2015.

[WO20] Andrew Wintenberg and Necmiye Ozay. "Implicit Invariant Sets for High-Dimensional Switched Affine Systems." In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3291–3297. IEEE, 2020.

[ZGK20] Changjian Zhang, David Garlan, and Eunsuk Kang. "A Behavioral Notion of Robustness for Software Systems." In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, pp. 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[ZR14] Yang Zhao and Kristin Yvonne Rozier. "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System." *Sci. Comput. Program.*, **96**(P3):337–353, December 2014.

[ZS14] D. Zhou and M. Schwager. "Vector field following for quadrotors using differential flatness." *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6567–6572, 2014.