

ROBUSTNESS OF TEMPORAL LOGIC SPECIFICATIONS

Georgios E. Fainekos

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2008

George J. Pappas Supervisor of Dissertation

Rajeev Alur Graduate Group Chair

In memory of my Grandmother

Who left this world just a couple of months
before her greatgrandson arrived.

Acknowledgements

I am extremely grateful to my adviser George J. Pappas for not only giving me the chance to study theoretical computer science (a notable risk if someone considers my hard core mechanical engineering background), but also for being a great mentor. George never micromanaged my research and he gave me the liberty to explore and define my own research goals. His extensive knowledge on almost any research subject and his vision helped me to avoid research deadlocks and to maximize my research output. I suspect that this dissertation might not have been possible without George's excellent proposal writing skills which led to my supporting funds NSF EHS 0311123, NSF ITR 0324977 and ARO MURI DAAD 19-02-01-0383.

My gratitude extends to my thesis committee members Rajeev Alur, Edmund M. Clarke, Insup Lee and Oleg Sokolsky for their comments and suggestions. Beyond my dissertation research, my interactions with Rajeev, Insup and Oleg through various projects and discussions have greatly helped me to shape my own research identity.

I would like to acknowledge my collaborators and friends Antoine Girard and Hadas Kress-Gazit on work that has appeared in this thesis. In particular, Chapters 6 and 7 would have not been possible in the current form without Antoine's fundamental contributions on the theory of approximate simulation relations. My interactions with Hadas have been crucial in developing the framework of temporal logic motion planning in Chapter 7. Over the years, I had the opportunity to co-author papers

which do not appear in this thesis. My appreciation goes to my co-authors and friends Madhukar Anand, Selcuk Bayraktar, A. Agung Julius and Savvas G. Loizou.

At the University of Pennsylvania I have found a truly magnificent research and intellectual environment. In the past, I have had the opportunity to engage into many stimulating discussions about theoretical and practical problems with Ali Ahmadzadeh, Stanislav Angelov, Ben Grocholsky, Boulos Harb, Jim Keller, Nader Motee, Paulo Tabuada and Hakan Yazarel. A special thanks goes to Jean Gallier. All these years, he has been a great teacher of mine and also my favorite next-door philosopher-mathematician. Furthermore, he attended both my proposal and my dissertation defense! I would also like to thank Michael Felker, our graduate student coordinator, for his prompt help with all my administrative related issues.

This dissertation is the outcome of my six year stay at the GRASP Laboratory at the University of Pennsylvania. Six years is a long time to survive at a place without good friends (and also beer, movies, philosophy, parties, sports etc). Besides the people I have mentioned above, my appreciation goes to Adam (a true philosopher and a great source of baby equipment), Hemantha (if for no other reason, for introducing me to my wife), Kilian (for our bizarre discussions in our office), Michalis (my conference travel buddy), Nima (for just being Nima), Stephen (you still owe me help for moving), Vasilis (for being an amazing roommate and friend for so many years) and, also, Ameesh, Anne, Arvind, Aveek, Bert, Calin, Christie, Dimitris, Evren, Fei, Joao, John, Kathryn, Lorenzo, Mirko, Nicolo, Ryan, Ted, Volkan and many others (for making Penn and Philly a fun place to be during my pre-fatherhood years).

My thought always goes back to Greece to my parents, Eugenios and Aikaterini, and my brother, Orestis. My parents' unconditional love and support and exemplary life have been essential to my academic success. All these years, I have struggled to attain my father's diverse and extraordinary engineering skills. I am still not close

enough. Even though I am grateful to all the members of my broader family for their encouragement and support, I would like to particularly acknowledge my uncle Nikos and aunt Sofia for all they have done for me.

Finally, I would like to thank the two most important people in my life, my wife Naomi and my son Leonidas. Naomi's optimism and positive attitude to life has almost eradicated my inherent pessimism and worst case scenario analysis of life events. I am also grateful to Naomi for tolerating the busy periods of my graduate student life. This last year - after my son was born - has been the most exhausting and at the same time happiest year of my life. I am glad that I was able to experience fatherhood while still a student. Thank you guys for making every single day of my life so unique!

ABSTRACT

ROBUSTNESS OF TEMPORAL LOGIC SPECIFICATIONS

Georgios E. Fainekos

Supervisor: George J. Pappas

Temporal logic verification has been proven to be a successful tool for the analysis of software and hardware systems. For such systems, both the models and the logic are Boolean valued. In the past, similar successful results have been derived for timed and linear hybrid systems. Even though the states of these systems are real valued, temporal logics are still interpreted over Boolean signals that abstract away the actual values of the real-valued state variables.

In this thesis, we advocate that in certain cases it is beneficial to define multi-valued semantics for temporal logics. That is, we consider a robust interpretation of Metric Temporal Logic (MTL) formulas over signals that take values in metric spaces. For such signals, which are generated by systems whose states are equipped with nontrivial metrics, for example continuous or hybrid, robustness is not only natural, but also a critical measure of system performance. The proposed multi-valued semantics for MTL formulas captures not only the usual Boolean satisfiability of the formula, but also topological information regarding the distance from unsatisfiability. This, in turn, enables the definition of robustness tubes that contain signals with the same temporal properties.

The notion of robustness for MTL specifications can be applied to at least 3 important problems. The first problem is the verification of continuous time signals with respect to MTL specifications using only discrete time analysis. The motivating idea behind our approach is that if the continuous time signal fulfills certain conditions and the discrete time signal robustly satisfies the MTL specification, then the correspond-

ing continuous time signal should also satisfy the same MTL specification. Second, the proposed robustness framework can be applied to the problem of bounded time temporal logic verification of dynamical systems. Our methodology has the distinctive feature that enables the verification of temporal properties of a dynamical system by checking only a finite number of its (simulated) trajectories. The interesting and promising feature of this approach is that the more robust the system is with respect to the temporal logic specification, the less is the number of simulations that are required in order to verify the system. Finally, the proposed definition of robustness for temporal logic specifications can be applied to the problem of automatic synthesis of hybrid systems. In particular, we address the problem of temporal logic motion planning for mobile robots that are modeled by second order dynamics. Temporal logic specifications can capture the usual control specifications such as reachability and invariance as well as more complex specifications like sequencing and obstacle avoidance. The resulting continuous time trajectory is provably guaranteed to satisfy the user specification.

Contents

1	Introduction	1
1.1	Systems' Design	1
1.2	Motivation and Contributions	7
1.2.1	System Analysis	7
1.2.2	Motion Planning for Mobile Robots	11
I	On Robustness	14
2	Linear Time Temporal Logics	15
2.1	Signals in Metric Spaces	15
2.1.1	Elements of Metric Spaces	15
2.1.2	Continuous-Time Signals	17
2.1.3	Discrete-Time Signals and Timed State Sequences	18
2.2	Metric & Linear Temporal Logic	20
2.2.1	Syntax	20
2.2.2	Continuous-Time Semantics	22
2.2.3	Discrete-Time Semantics	24
2.2.4	Negation Normal Form	27

3	Robustness of MTL Specifications over Signals	28
3.1	Continuous Time	28
3.1.1	Robustness Degree for Continuous-Time Signals	28
3.1.2	Robustness Estimate for Continuous-Time Signals	31
3.2	Discrete Time	39
3.2.1	Robustness Degree for Timed State Sequences	39
3.2.2	Robustness Estimate for Timed State Sequences	41
3.2.3	Testing the Robustness of Temporal Properties	43
3.3	TALIRO	47
3.3.1	Explanation of Input Arguments	47
3.3.2	Examples	52
3.4	Related Research and Future Work	56
4	From Signals to Systems	59
4.1	Dynamical Systems	59
4.2	Approximate (Bi)Simulation Relations	67
4.3	Robustness of MTL Specifications for Systems	72
4.3.1	Continuous Time	72
4.3.2	Discrete Time	74
II	Applications	75
5	Continuous-Time Satisfiability by Discrete-Time Reasoning	76
5.1	Introduction and Problem Formulation	76
5.2	Bounds on the Signal Values	77
5.3	Sampling for MITL Satisfiability	78
5.4	Sampling for MTL Robustness	83

5.5	Examples	87
5.6	Related Research	92
5.7	Conclusions and Future Work	92
6	Dynamical System Verification using Robust Simulations	94
6.1	Introduction and Problem Formulation	94
6.2	Approximating LPV Systems	98
6.3	MTL Robust Testing of LTI Systems	101
6.3.1	Sampling Using Bisimulation Functions	101
6.3.2	Verification Algorithm	103
6.4	Putting Everything Together	110
6.5	Related Research	114
6.6	Conclusions and Future Work	115
7	Temporal Logic Motion Planning	117
7.1	Introduction and Problem Formulation	117
7.2	Tracking using Approximate Simulation	121
7.3	Robust Interpretation of LTL Formulas	125
7.4	Temporal Logic Motion Planning	127
7.4.1	Discrete Abstraction of Robot Motion	128
7.4.2	Linear Temporal Logic Planning	130
7.4.3	Continuous Implementation of Discrete Trajectory	136
7.5	Putting Everything Together	140
7.6	Related Research	144
7.7	Conclusions and Future Work	146

III	Appendix	147
8	Proofs of Part I	148
8.1	Proofs of Section 3.1	148
8.1.1	Proof of Theorem 3.1.1	148
8.1.2	Proof of Proposition 3.1.2	153
8.1.3	Proof of Theorem 3.1.2	155
8.1.4	Proof of Proposition 3.1.3	156
8.1.5	Proof of Corollary 3.1.2	157
8.2	Proofs of Section 3.2	158
8.2.1	Recursive Formulation of Until in Section 3.2.3	158
8.2.2	Proof of Lemma 3.2.2	159
8.3	Proofs of Section 4.2	160
8.3.1	Proof of Theorem 4.2.3	160
9	Proofs of Part II	161
9.1	Proofs of Chapter 5	161
9.1.1	Proof of Proposition 5.3.1	161
9.1.2	Proof of Lemma 5.3.1	162
9.1.3	Proof of Lemma 5.3.2	163
9.1.4	Proof of Theorem 5.3.1	163
9.1.5	Proof of Lemma 5.4.2	164
9.1.6	Proof of Theorem 5.4.1	165
9.2	Proofs of Chapter 6	173
9.2.1	Proof of Theorem 6.2.1	173
9.2.2	Proof of Proposition 6.2.1	174
9.2.3	Proof of Theorem 6.3.1	175

9.2.4	Proof of Proposition 6.3.1	176
9.2.5	Proof of Theorem 6.3.2	177
9.3	Proofs of Chapter 7	177
9.3.1	Proof of Proposition 7.2.1	177
9.3.2	Proof of Theorem 7.2.1	178
9.3.3	Proof of Proposition 7.2.2	178
9.3.4	Proof of Proposition 7.4.1	181
9.3.5	Proof of Theorem 7.4.1	181
Nomenclature		185
Bibliography		195

List of Tables

3.1	Correspondence between logical operators and ASCII symbols.	49
3.2	Computation time for formula $\square (p1 \rightarrow \langle \rangle_{(0.0,1.0)} !p1)$	55
3.3	Computation time for formula (3.12) for the first row and for formula $\square_{[0.0,T]} (\langle \rangle_{[0.0,6.28]} (p2 / \langle \rangle_{[0.0,3.14]} p1))$ for the rest of the rows, where T is the maximum signal time minus 3π	55
6.1	Experimental results of the verification algorithm for the transmission line example. For each value of (T, θ) the table gives whether the property ψ_c holds on Σ_c^{LTI} and how many simulations of the system were necessary to conclude.	110
6.2	Approximation bounds for the Problems of Example 6.4.1.	112
6.3	Verification results for Example 6.4.1.	113

List of Figures

1.1	Two signals s_1 and s_2 which satisfy the specification: $\square(p_1 \rightarrow \diamond_{\leq 2} p_2)$. Here, p_1 labels the set $(-\infty, -10]$ and p_2 labels the set $[10, +\infty)$	8
1.2	The signal s_2 modified by random noise. The arrow points to the point in time where the property fails.	8
2.1	A continuous-time signal s_1 with time domain $R = [0, 7\pi]$	18
2.2	A discrete-time signal $\sigma_1(i) = \sin \tau_1(i) + \sin 2\tau_1(i)$ where the timing function is $\tau_1(i) = 0.2i$	18
3.1	The definition of distance and depth and the associated neighborhoods. Also, a tube (<i>dashed lines</i>) around a nominal signal s (<i>dash-dotted line</i>). The tube encloses a set of signals (<i>dotted lines</i>).	29
3.2	The signal s_1 and its robustness neighborhood with radius 0.2398. . .	33
3.3	On the left appears the time-domain representation of the discrete-time signals σ_1 and σ_2 of Example 3.2.1. On the right appears the space of the discrete-time signals of length 2. Each x represents a signal as a point in \mathbb{R}^2	41
3.4	The signal s_2	57
3.5	The samples of signal σ_1 are denoted by circles (o), while the samples of signal σ_2 by crosses (+).	57

3.6	The discrete-time signal $\sigma_3(i) = \sigma_1(i) - \sigma_2(i) $	57
4.1	Example 4.1.1 : the observation trajectory $y(t)$ with respect to time.	64
4.2	Example 4.1.1 : the observation trajectory $y(t)$ in phase space.	64
4.3	A ladder network representing a transmission line with 5 sections.	65
4.4	The matrix A_b of Example 4.1.2.	66
4.5	The matrix C of Example 4.1.2.	66
4.6	Example 4.1.2 : the observation trajectory $y^{(5)} = x^{(10)}$ with respect to time.	67
5.1	The sampled signal σ_2 generated by sampling the continuous-time signal $s_2(t) = \sin(t) + \sin(2t) + w(t)$, where $ w(t) \leq 0.1$, with constant sampling period 0.5. In this case, it is $ s_2(t_1) - s_2(t_2) \leq L_{s_1} t_1 - t_2 + w(t_1) + w(t_2) $. Thus, $\mathcal{E}_2(t) = L_{s_1}t + 0.2$	89
5.2	The output signal s_3 of Example 5.5.3.	91
6.1	1st iteration of the algorithm. The green rectangle indicates the set of initial conditions in the output space. The ellipsoid indicates the level set of the bisimulation function for $\varepsilon = 0.2924$ and the enclosing circle has radius $\varepsilon = 0.2924$. The plotted trajectory has robustness estimate 0.3852 (blue circle), while $\delta_a + \varepsilon = 0.5049$ (dashed red circle).	106
6.2	2nd iteration of the algorithm. The new hyper-rectangles and their respective centroids that result from the refinement process of X_a^0 with respect to x^0 . The circles indicate testing points that initiate trajectories that satisfy ψ_a δ_a -robustly, while the stars indicate testing points that must be refined locally.	107
6.3	The 10 simulations that verify that the system is at least 0.2125-robust with respect to specification ψ_a : Phase space.	108

6.4	The 10 simulations that verify that the system is at least 0.2125-robust with respect to specification ψ_a : Time domain.	108
6.5	An example trace of the RLC model of the transmission line.	109
6.6	The matrix $A'_b(p)$ of Example 6.4.1.	112
6.7	The 33 trajectories that are necessary for the verification of Problem Instance 2 of Example 6.4.1. This is the phase space for the variables $y^{(3)} - y^{(4)} - y^{(5)}$. The box indicates the set of initial conditions.	113
7.1	The simple environment of Example 7.1.1. The four regions of interest p_1, p_2, p_3, p_4 are enclosed by the polygonal region labeled by p_0	119
7.2	Hierarchical architecture of the hybrid controller H_ϕ	121
7.3	The modified workspace of Example 7.3.1 for $\delta = 1 + \varepsilon$ with ε sufficiently small.	126
7.4	Convex cell decomposition (Example 7.4.1).	130
7.5	(a) Reachability and (b) Cell invariant controller.	137
7.6	A trajectory of system Σ' for the path of Example 7.4.2 using the potential field controllers of [43].	140
7.7	The trajectory of system Σ which corresponds to the trajectory of system Σ' presented in Fig. 7.6.	142
7.8	Modified workspace for $\nu = 2.7$, i.e., $\delta = 5.4$	142
7.9	The initial environment of Example 7.5.2 and the resulting trajectory $x(t)$ of the dynamic robot Σ	143
7.10	The modified environment of Example 7.5.2, its triangulation and the trajectory $z(t)$ of the kinematic model Σ'	143
7.11	The distance between the trajectories $x(t)$ and $z(t)$	143

Preface

This dissertation is the outcome of my six year stay at the Department of Computer and Information Science at the University of Pennsylvania. When I started writing this thesis about one year ago, I was afraid that it might not be a coherent monograph, but rather a collection of papers. Nevertheless, the outcome has proved me wrong.

Chapter 1 gives a general introduction to system design and the related literature. The main goal of this chapter is to provide the motivation behind this thesis and to frame it within the existing research initiatives.

Part I is split into three chapters. Chapter 2 introduces Metric Temporal Logic (MTL) with Boolean semantics. A separate treatment of continuous and discrete time semantics is mandatory since the respective models cannot be unified in a way that maintains timing information in all cases. The fundamental contribution of this thesis, namely the robustness of a signal with respect to an MTL specification, is presented in Chapter 3. Chapter 4 extends the results of Chapter 3 from signals to systems. In this chapter, I also provide an overview of the theory of approximate simulation relations that will be necessary in Part II.

Part II discusses three applications of the robustness framework that was introduced in Part I. Chapter 5 introduces ways to infer the satisfiability of a continuous time signal using discrete time methods. This is done for two fragments of the logic MTL. Chapter 6 presents a method for the bounded time MTL verification of linear

systems with parametric uncertainties. Finally, Chapter 7 describes a method for the synthesis of hybrid controllers for motion planning of mobile robots from high level Linear Temporal Logic (LTL) specifications.

How to read this dissertation. The material in this thesis has been organized in such a way that the optimal ordering for reading the chapters is linear. However, the chapters in Part II do not have any dependencies among them. The reader is encouraged to read Part I before she attempts to read any of the chapters in Part II. If this is not possible, I have included a list of symbols (with page references) before the Bibliography that will ease the tedious task of directly attempting to decipher any of the chapters in Part II.

A last word of caution. Despite the fact that most of the material in this dissertation has been reviewed by numerous referees and collaborators of mine, every time I read it I still find some typos or mistakes. I would appreciate it you could report any new ones to : *fainekos at alumni.upenn.edu*.

Happy reading!

Georgios E. Fainekos
New York City, July 2008

This on-line version of my thesis contains some corrections over the published version in Chapter 3, which I discovered while working on the journal version of chapter.

Georgios E. Fainekos
New York City, September 2008

Chapter 1

Introduction

1.1 Systems' Design

Is automation the outcome of human ingenuity and constant struggle for improvement or simply the result of our lazy nature? Whatever the reason, automation is now an indispensable part of our lives. Both simple systems, such as a toilet flush, and complex ones, such as a power grid or an aircraft, involve automatic control procedures which enable the systems to perform tasks for us. The ubiquitous presence and, in most of the cases, the critical nature of automatic control systems mandates a disciplined scientific approach to their design.

However, not all control problems are of the same nature. Consider for example three simple, but very distinct systems : an industrial furnace, a mutual exclusion protocol and an electric heater. Industrial furnaces must monitor and maintain the pressure in the furnace at a certain value. The pressure in the chamber is regulated through a damper that releases gasses through a chimney to the atmosphere. A controller takes as input the pressure measurement from the chamber and outputs a signal that changes the position of the damper accordingly. In this case, the sensor

(pressure gauge), the actuator (damper) and the controller are analog devices. The heat process evolves in continuous time and the input and output signals are real valued. Such systems are usually referred to as *continuous-time dynamical systems* and can be modelled using differential equations. The study of continuous-time (or analog) control systems is a very mature field of study [148, 111] which traces its origins at least back to the eighteenth century when James Watt invented the centrifugal governor for the speed control of steam engines [140].

The advent of digital computers revolutionised the way we automatically control physical systems. Already from the early 90's microprocessors (MP) and digital signal processors (DSP) had replaced virtually all the analog controllers in industrial and safe-critical settings [120]. Consider again the furnace example. Now, the continuous-time signal from the pressure sensor is converted at an analog-to-digital (AD) converter into a discrete-time signal, which is in turn transmitted to the controller. The analog controller has now been replaced by a digital controller which processes the input discrete-time signal. The controller outputs a discrete-time signal which is converted into an analog signal that controls the damper. These types of systems are classified under the broad category of *discrete-time dynamical systems* and can be modelled (or approximated) by difference equations.

A mutual exclusion protocol can also be regarded as a (distributed) control system in a broader sense. Assume that we have a number of processes that run in parallel and that these processes need to access a shared resource. In order to guarantee consistency in the data of the system, we have to make sure that the processes do not interfere with each other when they access the resource for reading or writing. The goal of the mutual exclusion protocol is to coordinate the processes and to grant mutually exclusive access to a shared resource without assuming atomicity in the access method. Each instance of the protocol for each process, as well as the interaction of

the processes, can be modeled using finite state machines or automata [101]. Such systems fall within the broader class of *discrete event systems* [31].

Last, but not least, consider a simple home electric heater. The system consists of a radiator and a thermostat. The thermostat has two temperature settings; a low and a high temperature set point. When the temperature in the room is below the low temperature set point, the radiator turns on, while when the temperature in the room is higher than the high temperature set point, the radiator turns off. This system is intrinsically hybrid in nature : it contains a high level discrete part that governs the on-off operation of the radiator and it also contains a continuous-time process that describes the change of temperature in the room. The electric heater is an example of the relatively new class of *hybrid systems* [13, 8].

Whatever the class of systems may be, the goal of the engineer always remains the same : a successful system design. The success of the system design is judged by whether it fulfills a set of design-performance requirements (subject to economic and engineering constraints). The exact type of the design requirements depends on the class of the system and, of course, on the particular application at hand. Typical design requirements for continuous and discrete-time dynamical systems are stability (does the system state converge to the desired operating point?), error attenuation, rise time, maximum overshoot and others (see [120, 148]). The design requirements for discrete event systems are quite different. Typical examples are safety (is a set of undesirable system states always avoidable?), absence of non-progress cycles and deadlocks and more general requirements that can be expressed using temporal or modal logics [136, 56, 22]. In brief, temporal or modal logics can be used for reasoning about the properties of a system with respect to time. For example, using temporal logics, one can formally state specifications like : “Whenever a lock is acquired, then it is eventually released” or “There exists a computation path where p and q eventually

become true at the same time”. The most popular temporal logics for expressing design requirements for discrete event systems are the Linear Temporal Logic (LTL) [156] and the Computation Tree Logic (CTL) [40]. As one might expect, the design requirements for hybrid systems are usually a combination of the aforementioned specifications for dynamical and discrete event systems.

After the design requirements have been formally defined, there are mainly two ways to proceed with the system design.

- **Design/Analysis** : The engineer (or a team of engineers) comes up with an initial system design based on similar problems, her previous experience and her ingenuity. Then, the analysis part follows, where the engineer tries to determine if the system satisfies the design requirements. The analysis is usually performed in two ways : (i) a rigorous mathematical analysis is performed on a mathematical model of the system and/or (ii) certain adverse conditions are tested against a prototype or a mathematical model of the system. If the system, does not meet the design specifications, then the system is modified and the design/analysis cycle is repeated until a design that conforms to the design requirements has been achieved.
- **Synthesis** : This includes any design methodology that (semi-)automatically builds a system starting from the design requirements and other input constraints. The main advantage of any synthesis procedure is that the final system is guaranteed to meet the design requirements by construction.

When the analysis of the model of the system proves the correctness of the system with respect to the design specifications under any possible operating condition, e.g., presence of noise, set of initial conditions, uncertainty in model parameters, etc, then the analysis will be referred to as *system verification*. On the other hand, *testing* -

as the name implies - refers to the procedure of simulating the model (or executing the prototype) of the system using a particular selection of operating conditions. Thus, system verification can guarantee system correctness, while testing is mainly employed for the detection of mistakes in the design of the system.

The design-analysis cycle is usually the preferred way for designing a system in an industrial setting for both continuous [148, 167, 111] and discrete-time dynamical systems [120]. The asymptotic properties (e.g., stability) of these class of systems can be formally verified, while the transient properties (e.g., timing requirements) - in most of the cases - have to be tested for. For certain subclasses of dynamical systems, the problem of controller synthesis has been solved with tremendous success, e.g., optimal control [24] and robust control [55], leading to several industrial applications. Recently, the analysis of non-traditional control specifications, such as reachability (which is however related to the standard notion of controllability [148]), has attracted the attention of several researchers [36, 122, 179, 76, 46].

The situation is similar in the case of discrete event systems. Especially in the case of finite state machines most of the design requirements and in particular temporal logic specifications can be efficiently verified [41, 168]. Within the field of computer science, the research on formal verification has lead to several software toolboxes like SMV [141, 38], which can verify both CTL and LTL specifications, and SPIN [100], which can verify LTL specifications. Both software toolboxes have successfully detected bugs in existing software [91], protocol and hardware designs [39, 89]. Despite the successes of formal verification, the software engineers in industry still rely heavily on testing methodologies [17, 23] for the analysis of software [86]. On the synthesis front of discrete event systems, the numerous theoretical results on controller synthesis [175, 31], distributed system synthesis [158, 30], temporal logic synthesis [137, 121, 9, 154] and other [30] have not yet been met by an equivalent advance in technology

(software toolboxes).

Loosely speaking, some of the methods for the design of hybrid systems are a blend of the design methodologies for dynamical and discrete event systems. A lot of research (for a survey see [49]) has targeted the stability analysis of hybrid systems and, especially, of switched systems [126] - a subclass of hybrid systems. Great success in the verification of hybrid systems has been achieved through the reduction of certain classes of hybrid systems to discrete event systems [3]. In detail, the theories of timed automata [4] and linear hybrid automata [94] have led to the development of several verification toolboxes such as UPPAAL [20], HyTech [94] and PHAVer [69]. UPPAAL can verify specifications from a fragment of Timed CTL [1], while HyTech can verify Integrator CTL properties [7]. Due to the well-known undecidability results concerning the verification of hybrid systems [95, 98], many researchers have focused their efforts on the reachability (bounded time or without guarantees for termination) [37, 143, 14, 87, 123] and/or safety [159] problem of hybrid systems. This line of research has led to several software toolboxes : CheckMate [169], d/dt [45], Ellipsoidal Toolbox [123] and A Toolbox of Level Set Methods [144]. Since reachability methods for hybrid systems are computationally expensive, a lot of recent research effort has been targeted to the formal/systematic testing of hybrid systems [57, 173, 109, 155]. Hybrid systems' synthesis is also an active area of research. A lot of work has been devoted to the synthesis of timed automata [15, 176, 135, 19] and real time controllers [16]. The synthesis of hybrid automata from high level specifications has also been studied by several authors [132, 68, 115, 112, 145, 172, 85].

1.2 Motivation and Contributions

Up to this point, we have provided a very general description of the fundamental approaches in system design. The brief historical overview of the related research is not meant to be all inclusive, but to give a sense of which areas of system design are scientifically mature and where there is active exploration. In this section, we will discuss the motivation behind our work and highlight our contributions and their importance.

1.2.1 System Analysis

As mentioned in the previous section, temporal logic verification [41, 100] has been proven to be a very useful tool for the analysis of software and hardware systems. Even though temporal logic verification is possible for timed [4] and linear hybrid automata [94], the same is not true for dynamical or hybrid systems with more complex dynamics [95]. In practice, the validation of such systems still relies heavily on methods that involve systematic testing [109, 57]. More recently, temporal logic testing [134, 173] has been proposed as a framework that can provide us with additional information about the properties of continuous or discrete-time signals.

However, the classical approaches to the temporal logic testing involve a Boolean abstraction of the value of the signal with respect to the atomic propositions in the formula. This loss of information can be quite critical when we consider systems that model or control physical processes. For example, consider the signals s_1 and s_2 in Fig. 1.1. Both of them satisfy the same specification “if the value of the signal drops below -10, then it should also raise above 10 within 2 time units”. Nevertheless, a visual inspection of Fig. 1.1 indicates that there exists a qualitative difference between s_1 and s_2 . The later “barely” satisfies the specification. Indeed as we can see in Fig.

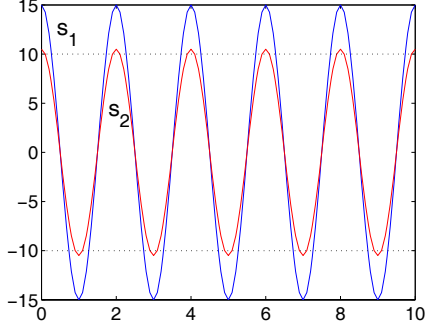


Figure 1.1: Two signals s_1 and s_2 which satisfy the specification: $\Box(p_1 \rightarrow \Diamond_{\leq 2} p_2)$. Here, p_1 labels the set $(-\infty, -10]$ and p_2 labels the set $[10, +\infty)$.

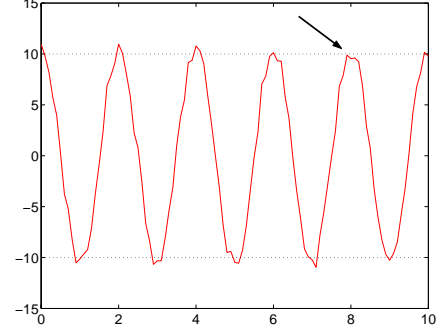


Figure 1.2: The signal s_2 modified by random noise. The arrow points to the point in time where the property fails.

1.2, adding a bounded noise on s_2 renders the property unsatisfiable on s_2 .

In order to differentiate between such trajectories of a system, in [64, 66] we introduced the concept of *robustness degree*. Informally, the robustness degree is the bound on the perturbation that the signal can tolerate without changing the truth value of a specification expressed in the Linear [156] or Metric Temporal Logic [116]. In detail, we consider signals which take values in some set X equipped with a metric d . If the time domain of these signals is R , then we can consider each signal as a point in X^R , which is the space of all possible signals with time domain R . In order to quantify how close are two different signals in X^R , we define the notion of distance using a metric ρ on the space X^R . Given an LTL or MTL formula ϕ , we can partition the space X^R into two sets: the set $\mathcal{L}(\phi)$ of signals that satisfy ϕ and the set $\mathcal{L}(\neg\phi)$ of signals that do not satisfy ϕ . Then, the formal definition of the robustness degree comes naturally, it is just the distance of a signal $s \in \mathcal{L}(\phi)$ from the set $\mathcal{L}(\neg\phi)$. Using the degree of robustness and the metric ρ , we can define an open ball (tube) around s and, therefore, we can be sure that any signal s' that remains within the open ball also stays in $\mathcal{L}(\phi)$. In this paper, we refer to such tubes as robust neighborhoods.

The robustness degree is not the only way to define robust neighborhoods. One can define multi-valued (or *robust* as it will be referred to in this thesis) semantics for MTL (or LTL) formulas. The atomic propositions in the robust version of MTL evaluate to the distance from the current value of the signal to the subset of X that the atomic proposition represents. As it is established in the framework of multi-valued logics [33, 51], the conjunction and disjunction in the Boolean logic are replaced by the inf and sup operations. Here, the logical negation is replaced by the usual negation over the reals. We prove that when an MTL formula is evaluated with robust semantics over a signal s , then it returns an under-approximation ε (*robustness estimate*) of the robustness degree and, therefore, any other signal s' that remains ε -close to s also satisfies the same specification.

Application-wise the importance of the robustness degree / estimate is immediate : if a system has the property that for near-by initial conditions (or under bounded disturbances etc) its signals remain δ -close to the nominal one and, also, its robustness degree / estimate with respect to an MTL formula ϕ is $\varepsilon > \delta$, then we know that all the system's signals also satisfy the same specification. This basic idea has been applied to the bounded time temporal logic verification of linear systems in [59]. In detail, besides the definition of robust satisfaction for MTL specifications, we also need the notion of *bisimulation functions* [82]. Bisimulation functions quantify the distance between two approximately bisimilar states and the trajectories initiating from them. Using a bisimulation function we can define a neighborhood of trajectories around a nominal one which have approximately the same behavior as the nominal trajectory. If this neighborhood of the simulated trajectory is contained in the tube of trajectories, which robustly satisfy the specification, then we can safely infer that the neighborhood of trajectories also satisfies the specification. Based on this observation, we develop an algorithm that, first, samples points in the set of initial conditions of

the system using guidance from the bisimulation function. Starting from this set of points, we simulate the system for a bounded horizon. For each of these trajectories we compute its robustness estimate. If the robustness estimate bounds the distance computed by the bisimulation function then we are done, otherwise we repeat the procedure. The novelty in our framework is that the number of simulations, which are required for the verification of the system, decreases inversely to the robustness of the system with respect to the temporal property.

Not surprisingly, bounded time temporal logic verification is not the only possible application for the robust semantics [65, 66]. Assume that we would like to test the transient response of an analog circuit to a predetermined input signal. Since analytical solutions exist only for a few simple cases, the design, verification and validation of such systems still relies heavily on testing the actual circuit or, more commonly, on simulations [153]. In either case, we end up with a discrete-time (or sampled) representation of the continuous-time signal that we have to analyze. On the other hand, the properties of the system that we would like to verify are – in most of the cases – with respect to the continuous-time behavior of the system. In particular, properties like overshoot, rise time, delay time, settling time and other constraints on the output signal [148] can be very naturally captured using Metric Temporal Logic (MTL) with continuous-time semantics [116]. The question that arises then is : Can we verify continuous-time properties of a system using only discrete-time reasoning?

We answer this question in the positive by deriving conditions on the dynamics of the signal and on the sampling function such that MTL reasoning over discrete-time signals can be applied to continuous-time signals. The main machinery that we employ for this purpose is the computation of the robustness estimate. All we need to do is to guarantee that the dynamics of the signal are such that between any two sampled points the actual continuous-time signal does not exceed the distance that is

computed using the robust semantics. The constraints on the sampling function play another role. They guarantee that there exist enough sampling points such that the validity of MTL formulas is maintained between the two different semantics [157].

1.2.2 Motion Planning for Mobile Robots

One of the main challenges in robotics is the development of mathematical frameworks that formally and verifiably integrate high level planning with continuous control primitives. Traditionally, the path planning problem for mobile robots has considered reachability specifications of the form “move from the Initial position I to the Goal position G while staying within region R ”. The solutions to this well-studied problem span a wide variety of methods, from continuous (like potential or navigation functions [35, §4]) to discrete (like Canny’s algorithm, Voronoi diagrams, cell decompositions and probabilistic road maps [35, 125]).

Whereas these methods solve the basic path planning problem, they do not address high level planning issues that arise when one considers a number of goals or a particular ordering of them. In order to manage such constraints, one should either employ an existing high level planning method [125] or attack the problem using optimization techniques like mixed integer linear programming [165]. Even though the aforementioned methods can handle partial ordering of goals, they cannot deal with temporally extended goals. For such specifications, planning techniques [74, 83] that are based on model checking [41] seem to be a better choice. Using temporally extended goals, one would sacrifice some of the efficiency of the standard planning methods for expressiveness in the specifications. Temporal logics such as Linear Temporal Logic (LTL) [156] have the expressive power to describe a conditional sequencing of goals under a well defined formal framework.

Such a formal framework can provide us with the tools for automated controller

synthesis and code generation. Beyond the provably correct synthesis of hybrid controllers for path planning from high level specifications, temporal logics have one more potential advantage when compared to other formalisms, e.g., regular languages [115]. That is to say, temporal logics were designed to bear a resemblance to natural language [149, §2.5]. Along the same lines, one can develop computational interfaces between natural language and temporal logics [124]. This fact alone makes temporal logics a suitable medium for our daily discourse with future autonomous agents. Toward this goal, we are working on the generation of hybrid motion planning controllers from inputs provided in formal English [117].

In [62, 61], we have combined such planning frameworks with local controllers defined over convex cells [21, 43] in order to perform temporal logic motion planning for a fully actuated kinematics model of a robot. In a kinematics model, the control inputs to the system are actually the desired velocities. However, when the velocity of the mobile robot is high enough, a kinematics model is not enough any more, necessitating thus the development of a framework that can handle a dynamics model. In a dynamics model, as opposed to a kinematics model, the control inputs are the forces (or accelerations) that act upon the system. In the synthesis part of this thesis, we provide a tractable solution to the LTL motion planning problem for dynamics models [58].

In order to solve this problem, we take a hierarchical approach. Our hierarchical control system consists of two layers. The first layer consists of a coarse and simple model of the plant (the kinematics model in our case). A controller is designed so that this abstraction meets the specification of the problem. The control law is then refined to the second layer, which in this paper consists of the dynamics model. Following [79], we first abstract the system to a kinematic model. An interface is designed so that the system is able to track the trajectories of its abstraction with

a given guaranteed precision δ . The control objective ϕ , which is provided by the user in LTL, is then modified and replaced by an LTL formula ϕ' in negation normal form which is also given a more robust interpretation. The formula ϕ' is such that given a trajectory satisfying ϕ' , any trajectory remaining within distance δ satisfies ϕ . It then remains to design a controller $H'_{\phi'}$ for the abstraction such that all its trajectories satisfy the “robust” specification ϕ' . This is achieved by first lifting the problem to the discrete level by partitioning the environment into a finite number of equivalence classes. A variety of partitions are applicable [35, 125], but we focus on triangular decompositions [53] and general convex decompositions [110] as these have been successfully applied to the basic path planning problem in [21] and [43] respectively. The partition results in a natural discrete abstraction of the robot motion which is then used for planning with automata theoretic methods [74]. The resulting discrete plan acts as a supervisory controller that guides the composition of local vector fields [21, 43] which generate the desired motion. Finally, using the hierarchical control architecture, we lift the hybrid controller $H'_{\phi'}$, which guarantees the generation of trajectories that satisfy ϕ' for the kinematics model, to a hybrid controller H_{ϕ} for the dynamics model which now generates trajectories that satisfy the original specification ϕ .

Part I

On Robustness

Chapter 2

Linear Time Temporal Logics

In this chapter, we define signals over metric spaces and provide a brief overview of Temporal Logics (TL) that are interpreted over linear time structures.

2.1 Signals in Metric Spaces

2.1.1 Elements of Metric Spaces

A metric space (X, d) is a set X with a metric d . For a short introduction to metric spaces see [146]. In this paper, we only use the notions of metric and neighborhood which we define below.

Definition 2.1.1 (Metric). *A metric on a set X is a positive function $d : X \times X \rightarrow \overline{\mathbb{R}}_{\geq 0}$, such that the following properties hold*

1. $\forall x_1, x_2 \in X, d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$
2. $\forall x_1, x_2 \in X, d(x_1, x_2) = d(x_2, x_1)$
3. $\forall x_1, x_2, x_3 \in X, d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$

In the definition above, \mathbb{R} is the set of the real numbers. We denote the extended real number line by $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. In addition, we use pseudo-arithmetic expressions to represent certain subsets of the aforementioned sets. For example, $\mathbb{R}_{\geq 0}$ denotes the subset of the reals whose elements are greater than or equal to zero.

Using a metric d , we can define the distance of a point $x \in X$ from a set $S \subseteq X$. Intuitively, this distance is the shortest distance from x to all the points in S . In a similar way, the depth of a point x in a set S is defined to be the shortest distance of x from the boundary of S .

Definition 2.1.2 (Distance, Depth, Signed Distance [26] §8). *Let $x \in X$ be a point, $S \subseteq X$ be a set and d be a metric on X . Then, we define the*

- *Distance from x to S to be $\mathbf{dist}_d(x, S) := \inf\{d(x, y) \mid y \in \overline{S}\}$*
- *Depth of x in S to be $\mathbf{depth}_d(x, S) := \mathbf{dist}_d(x, X \setminus S)$*
- *Signed Distance from x to S to be*

$$\mathbf{Dist}_d(x, S) := \begin{cases} -\mathbf{dist}_d(x, S) & \text{if } x \notin S \\ \mathbf{depth}_d(x, S) & \text{if } x \in S \end{cases}$$

The notation \overline{S} denotes the closure of a set S , that is, the intersection of all closed sets containing S . We should point out that we use the extended definition of supremum and infimum. In other words, the supremum of the empty set is defined to be bottom element of the domain, while the infimum of the empty set is defined to be the top element of the domain. For example, when we reason over $\overline{\mathbb{R}}$ as above, then $\sup \emptyset := -\infty$ and $\inf \emptyset := +\infty$. Also of importance is the notion of an open ball of radius ε centered at a point $x \in X$.

Definition 2.1.3 (ε -Ball). *Given a metric d , a radius $\varepsilon > 0$ and a point $x \in X$, the open ε -ball centered at x is defined as $B_d(x, \varepsilon) = \{y \in X \mid d(x, y) < \varepsilon\}$.*

The following properties of the ε -ball are immediate. Given $0 < \varepsilon < \varepsilon'$ and a point $x \in X$, we have $B_d(x, \varepsilon) \subseteq B_d(x, \varepsilon')$. Also, if the distance (\mathbf{dist}_d) of a point x from a set S is $\varepsilon > 0$, then $B_d(x, \varepsilon) \cap S = \emptyset$. Note that \mathbf{dist}_d actually returns the radius of the largest ball $B_d(x, \varepsilon)$ that fits in the set $X \setminus S$. Similarly, it is easy to see that if $\mathbf{depth}_d(x, S) = \varepsilon > 0$, then $B_d(x, \varepsilon) \subseteq S$.

2.1.2 Continuous-Time Signals

We use continuous-time signals in order to capture the behavior of real-time or physical systems. Typical models of real time systems are the formalisms of timed automata [4], hybrid automata [93] and dynamical systems [34, 111]. Formally, a *continuous-time signal* s is a map $s : R \rightarrow X$ such that R is the time domain and X is a *metric* space. When we consider bounded time signals, then $R = [0, r] \subseteq \mathbb{R}_{\geq 0}$ with $r > 0$, otherwise we let $R = \mathbb{R}_{\geq 0}$.

We use the notation $\mathfrak{F}(R, X)$ to denote the set of all possible signals from R to X . More generally, given two sets A, B , the set $\mathfrak{F}(A, B)$ denotes the set of all functions from A to B . That is, $\mathfrak{F}(A, B) = B^A$ and for any $f \in \mathfrak{F}(A, B)$, we have $f : A \rightarrow B$. The domain of a function $f \in \mathfrak{F}(A, B)$ is denoted by $\mathbf{dom}(f)$.

In the following, we fix R to refer to a time domain as described above. As an example of a continuous-time signal, consider the function $s_1(t) = \sin t + \sin 2t$ in Fig. 2.1 such that $R = [0, 7\pi]$.

A Boolean valued signal is a signal $s : R \rightarrow \mathbb{B}$, i.e., in this case $X = \mathbb{B}$. Here, $\mathbb{B} = \{\perp, \top\}$, where \top and \perp are the symbols for the boolean constants *true* and *false* respectively. Note that the metric d_d , where $d_d(x, y) = 1$ if $x \neq y$ and $d_d(x, y) = 0$

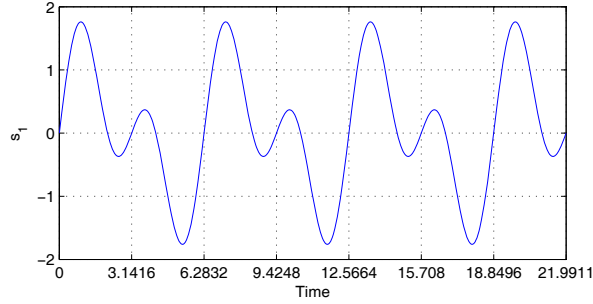


Figure 2.1: A continuous-time signal s_1 with time domain $R = [0, 7\pi]$.

otherwise, induces the discrete topology on \mathbb{B} .

2.1.3 Discrete-Time Signals and Timed State Sequences

A *discrete-time signal* σ can represent computer simulated trajectories of physical models or the sampling process that takes place when we digitally monitor physical systems. Informally, a discrete-time signal σ is a sequence of snapshots of the continuous-time behaviour of a system. Each such snapshot represents the state of the system at a particular point in time (see Fig. 2.2). However, a discrete-time signal does not provide us with any timing information. In order to reason about the timing properties of a discrete-time signal, we introduce the *timing function* τ . The role of the timing function is to pair each snapshot with a time stamp.

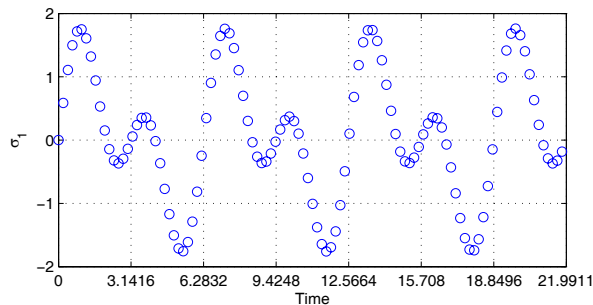


Figure 2.2: A discrete-time signal $\sigma_1(i) = \sin \tau_1(i) + \sin 2\tau_1(i)$ where the timing function is $\tau_1(i) = 0.2i$.

More formally, we define a discrete-time signal σ to be a function from the set

$\mathfrak{F}(N, X)$. Such a signal can be of bounded or unbounded duration. In the former case we set $N = \mathbb{N}_{\leq n}$ for some $n \in \mathbb{N}$, while in the latter $N = \mathbb{N}$. Here, \mathbb{N} is the set of the natural numbers. In the following, we fix $N \subseteq \mathbb{N}$ to be the domain of the discrete-time signal. Analogously, a timing function τ is a member of the set $\mathfrak{F}(N, \mathbb{R}_{\geq 0})$. Two important restrictions on a timing function τ are

1. τ must be a strictly increasing function, i.e., $\tau(i) < \tau(j)$ for $i < j$.
2. if N is infinite, then τ must diverge, i.e., $\lim_{i \rightarrow +\infty} \tau(i) = +\infty$.

We denote the set of strictly increasing functions from N to \mathbb{R} which diverge by $\mathfrak{F}^\dagger(N, \mathbb{R}) \subseteq \mathfrak{F}(N, \mathbb{R})$. Of particular interest to us are the timing functions for which the time difference between any two consecutive timestamps is constant. That is, for each timing function τ in this class there exists some constant $\alpha \in \mathbb{R}_{>0}$ such that $\tau(i) = \alpha i$ for $i \in N$. We will denote the set of such functions from N to \mathbb{R} by $\mathfrak{F}_c^\dagger(N, \mathbb{R}) \subseteq \mathfrak{F}^\dagger(N, \mathbb{R})$, where c stands for constant.

By pairing a discrete-time signal σ with a timing function τ , we define what is usually referred to as a *timed state sequence* $\mu = (\sigma, \tau)$, i.e., $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$. In the following, we let $\mu^{(1)}$ be the first member of the pair, i.e., $\mu^{(1)} = \sigma$, and $\mu^{(2)}$ be the second member of the pair, i.e., $\mu^{(2)} = \tau$. Notice that the pair $(\mathcal{O}^{-1} \circ \sigma, \tau)$ is actually a Boolean-valued timed state sequence, which is a widely accepted model for reasoning about real time systems [6, 150]. Here, \mathcal{O}^{-1} denotes a function that maps points of the space X to a set of atomic propositions (see Section 2.2.2) and \circ denotes function composition : $(f \circ g)(t) = f(g(t))$.

In some applications, we are interested in monitoring a continuous-time signal $s \in \mathfrak{F}(R, X)$. Since the monitoring process is achieved through a digital computer, we have to deal with the inherent discretization or sampling of the continuous-time signal. We can model the sampling process by assuming that there exists a timing

function τ which returns the time instants that we have sampled the continuous-time signal s . Thus, the sampled signal σ is simply the function composition of s with τ , i.e., $\sigma = s \circ \tau$. In this case, a timing function represents something more concrete. It returns the points in time at which we have sampled the continuous-time signal. Hence when sampling is involved, we will refer to the timing function as *sampling function* $\hat{\tau}$ and we will consider it is a member of the set $\mathfrak{F}^\dagger(N, R)$ instead of $\mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$.

2.2 Metric & Linear Temporal Logic

Metric Temporal Logic (MTL) was introduced in [116] as an extension to Linear Temporal Logic (LTL) [27, 156]. LTL is useful for reasoning about the qualitative properties of signals, e.g., sequences of events, whereas using MTL one can reason about the quantitative timing properties of signals, e.g., elapsed time between two events. In this section, we review the basics of the syntax and the semantics of these temporal logics.

2.2.1 Syntax

In the following definition, we first introduce the MTL syntax and, then, derive from that the LTL and MITL syntax. Metric Interval Temporal Logic (MITL) [5] is a restricted version of MTL where the timing intervals of the temporal operators are not allowed to be singleton sets.

Definition 2.2.1 (Syntax). *Let \mathbb{C} be the set of truth degree constants, AP be the set of atomic propositions and \mathcal{I} be any non-empty interval of $\overline{\mathbb{R}}_{\geq 0}$. The set $MTL_{\mathbb{C}}(AP)$ of all well-formed MTL formulas (wff) is inductively defined using the following gram-*

mar:

$$\phi ::= c \mid p \mid \neg\phi \mid \phi \vee \phi \mid \phi \mathcal{U}_{\mathcal{I}}\phi$$

where $c \in \mathbb{C}$ and $p \in AP$. If the rule $\neg\phi$ is replaced by $\neg p$ and we add the rules $\phi \wedge \phi \mid \phi \mathcal{R}_{\mathcal{I}}\phi$ to the grammar, then we say that the formula is in *Negation Normal Form (NNF)*. In this case, the set of wff is denoted by $MTL_{\mathbb{C}}^+(AP)$. The set $MTL_{\mathbb{C}}(AP, op_1, op_2, \dots)$ denotes the subset of MTL formulas that contain only the operators op_1, op_2, \dots . If $\inf \mathcal{I} < \sup \mathcal{I}$, then the set $MTL_{\mathbb{C}}(AP)$ reduces to the set of all well-formed MITL formulas which is denoted by $MITL_{\mathbb{C}}(AP)$. If $\mathcal{I} = [0, +\infty)$, then the set $MTL_{\mathbb{C}}(AP)$ reduces to the set of all well-formed LTL formulas which is denoted by $LTL_{\mathbb{C}}(AP)$.

In the above definition, $\mathcal{U}_{\mathcal{I}}$ is the timed *until* operator and $\mathcal{R}_{\mathcal{I}}$ the timed *release* operator. The subscript \mathcal{I} imposes timing constraints on the temporal operators. Informally, the formula $\phi_1 \mathcal{U}_{\mathcal{I}}\phi_2$ expresses the property that within the time interval \mathcal{I} from the current moment in time, there exists some time that ϕ_2 becomes true over the given signal and, furthermore, for all previous times (besides the current time), the signal satisfies ϕ_1 . Intuitively, the release operator $\phi_1 \mathcal{R}_{\mathcal{I}}\phi_2$ states that ϕ_2 should always hold during the interval \mathcal{I} , a requirement which is released when ϕ_1 becomes true. Syntactically, \mathcal{U} and \mathcal{R} are dual operators, that is, $\phi_1 \mathcal{U}_{\mathcal{I}}\phi_2 = \neg(\neg\phi_1 \mathcal{R}_{\mathcal{I}}\neg\phi_2)$ and $\phi_1 \mathcal{R}_{\mathcal{I}}\phi_2 = \neg(\neg\phi_1 \mathcal{U}_{\mathcal{I}}\neg\phi_2)$. When $\mathbb{B} \subseteq \mathbb{C}$, we can also define the temporal operators *eventually* $\diamond_{\mathcal{I}}\phi = \top \mathcal{U}_{\mathcal{I}}\phi$ and *always* $\square_{\mathcal{I}}\phi = \perp \mathcal{R}_{\mathcal{I}}\phi$.

The interval \mathcal{I} can be open, half-open or closed, bounded or unbounded, but it must be non-empty ($\mathcal{I} \neq \emptyset$). Moreover, we define the following operations on the timing constraints \mathcal{I} of the temporal operators

$$t + \mathcal{I} := \{t + t' \mid t' \in \mathcal{I}\} \quad \text{and} \quad t +_R \mathcal{I} := (t + \mathcal{I}) \cap R$$

for any t in R . Sometimes for clarity in the presentation, we replace \mathcal{I} with pseudo-metric expressions, e.g., $\mathcal{U}_{[0,1]}$ is written as $\mathcal{U}_{\leq 1}$. In the case where $\mathcal{I} = [0, +\infty)$, we remove the subscript \mathcal{I} from the temporal operators, e.g., we just write \mathcal{U} , \mathcal{R} , \square , \diamond .

2.2.2 Continuous-Time Semantics

In this section, MTL formulas are interpreted over continuous-time signals. The atomic propositions for the class of problems we are dealing with label subsets of the set X . In other words, we define an observation map $\mathcal{O} : AP \rightarrow \mathcal{P}(X)$ such that for each $p \in AP$ the corresponding set is $\mathcal{O}(p) \subseteq X$. Here, $\mathcal{P}(S)$ denotes the powerset of the set S .

In this thesis, we define the continuous-time Boolean semantics of MTL formulas using a valuation function $\langle\langle \cdot, \cdot \rangle\rangle_C : MTL_{\mathbb{B}}(AP) \times \mathfrak{F}(AP, \mathcal{P}(X)) \rightarrow (\mathfrak{F}(R, X) \times R \rightarrow \mathbb{B})$ and we write $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \top$ instead of the usual notation $(\mathcal{O}^{-1} \circ s, t) \models \phi$. Here, $\mathcal{O}^{-1} : X \rightarrow \mathcal{P}(AP)$ is defined as $\mathcal{O}^{-1}(x) := \{p \in AP \mid x \in \mathcal{O}(p)\}$ for $x \in X$. In this case, we say that the signal s under observation map \mathcal{O} satisfies the formula ϕ at time t . In the proofs, we drop \mathcal{O} from the notation for brevity only when this does not cause any confusion. We are therefore interested in checking whether $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, 0) = \top$. In this case, we refer to s as a *model* of ϕ and we just write $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s) = \top$ for brevity.

Before proceeding to the actual definition of the semantics, we introduce some auxiliary notation. If $(\mathbb{V}, <)$ is a totally ordered set, then we define the binary operators $\sqcup : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{V}$ and $\sqcap : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{V}$ using the supremum and infimum functions as $x \sqcup y := \sup\{x, y\}$ and $x \sqcap y := \inf\{x, y\}$. Also, for some $V \subseteq \mathbb{V}$ we extend the above definitions as follows $\bigsqcup V := \sup V$ and $\bigsqcap V := \inf V$. Again, we use the extended definition of the supremum and infimum, i.e., $\sup \emptyset := \inf V$ and $\inf \emptyset := \sup V$. Since $(\mathbb{V}, <)$ is a totally ordered set, it is also a *distributive lattice* (see

Example 4.6 (2) in [48]), i.e., for all $a, b, c \in \mathbb{V}$, we have $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$ and $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$. Note that the structure $(\mathbb{B}, <)$ is a totally ordered set with $\perp < \top$ and that $(\mathbb{B}, \sqcap, \sqcup, \neg)$ is a boolean algebra with the complementation defined as $\neg\top = \perp$ and $\neg\perp = \top$.

Definition 2.2.2 (Continuous-Time Semantics). *Let $\phi \in MTL_{\mathbb{B}}(AP)$ be an MTL formula, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ be an observation map and $s \in \mathfrak{F}(R, X)$ be a continuous-time signal, then the continuous-time semantics of ϕ is defined by*

$$\begin{aligned} \langle\langle \top, \mathcal{O} \rangle\rangle_C(s, t) &:= \top \\ \langle\langle p, \mathcal{O} \rangle\rangle_C(s, t) &:= K_{\in}(s(t), \mathcal{O}(p)) = \begin{cases} \top & \text{if } s(t) \in \mathcal{O}(p) \\ \perp & \text{otherwise} \end{cases} \\ \langle\langle \neg\phi_1, \mathcal{O} \rangle\rangle_C(s, t) &:= \neg\langle\langle \phi_1, \mathcal{O} \rangle\rangle_C(s, t) \\ \langle\langle \phi_1 \vee \phi_2, \mathcal{O} \rangle\rangle_C(s, t) &:= \langle\langle \phi_1, \mathcal{O} \rangle\rangle_C(s, t) \sqcup \langle\langle \phi_2, \mathcal{O} \rangle\rangle_C(s, t) \\ \langle\langle \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2, \mathcal{O} \rangle\rangle_C(s, t) &:= \bigsqcup_{t' \in (t +_R \mathcal{I})} (\langle\langle \phi_2, \mathcal{O} \rangle\rangle_C(s, t') \sqcap \prod_{t < t' < t''} \langle\langle \phi_1, \mathcal{O} \rangle\rangle_C(s, t'')) \end{aligned}$$

where $t, t', t'' \in R$ and K_{\in} is the characteristic function of the \in relation.

In the above definition, the until operator is quantified over the set $t +_R \mathcal{I}$ instead of $t + \mathcal{I}$ because we also consider signals of bounded duration.

We denote by $\mathcal{L}_t(\phi, \mathcal{O}) = \{s \in \mathfrak{F}(R, X) \mid \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \top\}$ the set of all signals that satisfy ϕ at time t . Then $\mathcal{L}(\phi, \mathcal{O}) = \mathcal{L}_0(\phi, \mathcal{O})$ is the set of all models of ϕ . We say that the formula ϕ is *valid* when $\mathcal{L}(\phi, \mathcal{O}) = \mathfrak{F}(R, X)$ and *invalid* when $\mathcal{L}(\phi, \mathcal{O}) = \emptyset$. Note that $\mathcal{L}_t(\neg\phi, \mathcal{O}) = \{s \in \mathfrak{F}(R, X) \mid \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \perp\}$ since $\langle\langle \neg\phi, \mathcal{O} \rangle\rangle_C(s, t) = \neg\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \top$. Therefore, the sets $\mathcal{L}_t(\phi, \mathcal{O})$ and $\mathcal{L}_t(\neg\phi, \mathcal{O})$ are complements of each other with respect to $\mathfrak{F}(R, X)$. Thus, $\mathfrak{F}(R, X) \setminus \mathcal{L}_t(\phi, \mathcal{O}) = \mathcal{L}_t(\neg\phi, \mathcal{O})$ and vice versa. Formally, in the notation $\mathcal{L}_t(\phi, \mathcal{O})$ we should have also specified the time

domain R and the metric space X of the signal. However, we feel that the addition of R and X in the notation would only make the notation more cluttered, while omitting them does not cause any confusion since the time domain and the metric space are always clear from the context.

Remark 2.2.1. *We conclude this section with a word of caution. Even though we allow in our definitions signals of unbounded duration, our logical framework cannot capture asymptotic properties with respect to time. For example, consider the signal $s(t) = \exp(-t)$ which converges to 0 as t goes to $+\infty$. This signal does not satisfy the specification $\diamond p$, where $\mathcal{O}(p) = (-\infty, 0]$ since there does not exist some time t such that $s(t) = 0$, i.e., $s(t) \in \mathcal{O}(p)$. Therefore, it is natural to consider bounded time domains since we cannot express asymptotic properties with MTL.*

2.2.3 Discrete-Time Semantics

Physical world processes evolve in real time and, hence, the requirements for such systems must be specified in continuous-time formalisms as well. However, in virtually all the practical cases the representation of the behavior of such systems that is available to us for analysis is in discrete time. For example, when we monitor the temperature in a room, we cannot know the value of the continuous-time signal at all points in time, but only at those points in time that are attainable through an analog-to-digital converter. This is also true, when we test, simulate or verify a continuous-time signal using a digital computer. Some form of discretization of time is always necessary.

The semantics of MTL as it was introduced in Definition 2.2.2 can actually be defined over signals whose domain is any linearly ordered time flow. Therefore, it is possible to define a signal over the natural numbers and perform discrete-time

temporal logic analysis over that. However, the timing constraints in this case refer to the number of samples taken from the continuous-time signal and not to the actual real-time constraints. When the sampling step is constant, then there exists a simple conversion between the number of samples and the time that they were taken. But it is not always the case that the sampling step is constant and, moreover, the user often needs to provide real-time requirements on the signal which refer to the actual evolution of time and not the number of samples. Hence, in this section we define MTL semantics over timed state sequences.

Again, the semantics is defined using a valuation function. Given a TSS μ , we write $\langle\langle\phi, \mathcal{O}\rangle\rangle_D(\mu, i) = \top$ when μ satisfies the formula ϕ at moment i . Similarly to the continuous-time case, when $i = 0$ and the formula evaluates to \top , then we refer to μ as a *model* of ϕ and we write $\langle\langle\phi, \mathcal{O}\rangle\rangle_D(\mu) = \top$.

In the definition below, we also use the following notation : for $S \subseteq \mathbb{R}_{\geq 0}$, the *preimage* of S under τ is defined as : $\tau^{-1}(S) := \{i \in N \mid \tau(i) \in S\}$.

Definition 2.2.3 (Discrete-Time Semantics). *Let $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$ and $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, then the discrete-time semantics¹ of any formula $\phi \in MTL_{\mathbb{B}}(AP)$ is defined recursively as follows*

$$\langle\langle\top, \mathcal{O}\rangle\rangle_D(\mu, i) := \top$$

$$\langle\langle p, \mathcal{O}\rangle\rangle_D(\mu, i) := K_{\in}(\sigma(i), \mathcal{O}(p))$$

$$\langle\langle \neg\phi_1, \mathcal{O}\rangle\rangle_D(\mu, i) := \neg\langle\langle\phi_1, \mathcal{O}\rangle\rangle_D(\mu, i)$$

$$\langle\langle\phi_1 \vee \phi_2, \mathcal{O}\rangle\rangle_D(\mu, i) := \langle\langle\phi_1, \mathcal{O}\rangle\rangle_D(\mu, i) \sqcup \langle\langle\phi_2, \mathcal{O}\rangle\rangle_D(\mu, i)$$

¹In Definition 2.2.2, the continuous-time semantics of until is strict, i.e., ϕ_1 does not have to hold at time t or t' , while here, the discrete-time semantics of until is non-strict. We should remark that the discrete-time robustness estimate can also be defined using strict semantics and that the non-strict semantics is preferred because it greatly simplifies the presentation of the material in Chapters 5 and 7.

$$\langle\langle\phi_1 \mathcal{U}_I \phi_2, \mathcal{O}\rangle\rangle_D(\mu, i) := \bigsqcup_{j \in \tau^{-1}(\tau(i)+I)} (\langle\langle\phi_2, \mathcal{O}\rangle\rangle_D(\mu, j) \sqcap \prod_{i \leq k < j} \langle\langle\phi_1, \mathcal{O}\rangle\rangle_D(\mu, k))$$

where $i, j, k \in N$, $\sigma = \mu^{(1)}$, $\tau = \mu^{(2)}$ and K_ϵ is the characteristic function of the \in relation.

We denote by $TSS_i(\phi, \mathcal{O}) = \{\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0}) \mid \langle\langle\phi, \mathcal{O}\rangle\rangle_D(\mu, i) = \top\}$ the set of all timed state sequences that satisfy ϕ at time i . Then, $TSS(\phi, \mathcal{O}) = TSS_0(\phi, \mathcal{O})$ is the set of all timed state sequences that are models of ϕ . In this work, we are not interested in all the discrete-time models of ϕ , but only in those that have the same timing function τ with the input timed state sequence μ . This is because we are not interested in studying the robustness of the input timed state sequence with respect to its timing constraints as it is done in [84, 105, 25], but with respect to the constraints imposed on the value of the signal by the atomic propositions. Thus, for a given timing function τ , we also define the set $TSS_i^\tau(\phi, \mathcal{O}) = \{\mu \in TSS_i(\phi, \mathcal{O}) \mid \mu^{(2)} = \tau\}$.

Since we only consider models with the same timing function, we can ignore the timing function altogether and use the corresponding discrete-time signal. Therefore, we also define the set $\mathcal{L}_i^\tau(\phi, \mathcal{O}) = \{\sigma \in \mathfrak{F}(N, X) \mid (\sigma, \tau) \in TSS_i(\phi, \mathcal{O})\}$. Since $\mu \notin TSS_i(\phi, \mathcal{O})$ if and only if $\mu \in TSS_i(\neg\phi, \mathcal{O})$, we also get that $\sigma \notin \mathcal{L}_i^\tau(\phi, \mathcal{O})$ if and only if $\sigma \in \mathcal{L}_i^\tau(\neg\phi, \mathcal{O})$ for $\sigma = \mu^{(1)}$. Hence, $\mathcal{L}_i^\tau(\neg\phi, \mathcal{O}) = \mathfrak{F}(N, X) \setminus \mathcal{L}_i^\tau(\phi, \mathcal{O})$.

When we only consider LTL formulas, our notation can be reduced to the notation used for continuous-time signals. The only thing that changes is the underlying time domain. However, since in Part II we will be explicitly using LTL with discrete-time semantics, we review here the simplified notation. Given a discrete-time signal σ , we write $\langle\langle\phi, \mathcal{O}\rangle\rangle_D(\sigma, i) = \top$ when σ satisfies the formula ϕ at time i . When $i = 0$ and the formula evaluates to \top , then we refer to σ as a *model* of ϕ and we write

$\langle\langle\phi, \mathcal{O}\rangle\rangle_D(\sigma) = \top$. The set $\hat{\mathcal{L}}_i(\phi, \mathcal{O}) = \{\sigma \in \mathfrak{F}(N, X) \mid \langle\langle\phi, \mathcal{O}\rangle\rangle_D(\sigma, i) = \top\}$ is the set of all discrete-time signals that satisfy ϕ at time i .

2.2.4 Negation Normal Form

In certain cases, it is beneficial to convert a logic formula into a normal form. In this thesis, we will extensively use the Negation Normal Form (NNF). In order to convert an MTL formula into NNF, we push the negations inside the subformulas such that the only allowed negation operators appear in front of atomic propositions. The following result is immediate from the Boolean semantics of MTL formulas.

Lemma 2.2.1. *Given $\phi \in MTL_{\mathbb{B}}(AP)$, the translation of ϕ to its equivalent formula in Negation Normal Form is achieved using the following rewriting rules*

$$\begin{aligned} \neg\neg\phi &= \phi \\ \neg(\phi_1 \vee \phi_2) &= \neg\phi_1 \wedge \neg\phi_2 & \neg(\phi_1 \wedge \phi_2) &= \neg\phi_1 \vee \neg\phi_2 \\ \neg(\phi_1 \mathcal{U}_{\mathcal{I}}\phi_2) &= \neg\phi_1 \mathcal{R}_{\mathcal{I}}\neg\phi_2 & \neg(\phi_1 \mathcal{R}_{\mathcal{I}}\phi_2) &= \neg\phi_1 \mathcal{U}_{\mathcal{I}}\neg\phi_2 \end{aligned}$$

We denote the function that applies the above rules to ϕ in a recursive way by \mathbf{nnf} , that is, $\mathbf{nnf} : MTL_{\mathbb{B}}(AP) \rightarrow MTL_{\mathbb{B}}^+(AP)$. Then, given $s \in \mathfrak{F}(R, X)$ and $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, we have $\langle\langle\phi, \mathcal{O}\rangle\rangle_C(s) = \langle\langle\mathbf{nnf}(\phi), \mathcal{O}\rangle\rangle_C(s)$ and $\langle\langle\phi, \mathcal{O}\rangle\rangle_D(\mu) = \langle\langle\mathbf{nnf}(\phi), \mathcal{O}\rangle\rangle_D(\mu)$.

Chapter 3

Robustness of MTL Specifications over Signals

This chapter introduces a new notion of robustness for signals with respect to temporal logic specifications. Our notion of robustness refers to robustness with respect to the value of the signal and not with respect to any possible timing constraints imposed by the formula. Since the models we use in continuous and discrete time differ, we must also define separately the continuous and discrete-time robustness notion.

3.1 Continuous Time

3.1.1 Robustness Degree for Continuous-Time Signals

In this section, we define what it means for a signal $s \in \mathfrak{F}(R, X)$ to satisfy a Metric Temporal Logic specification *robustly*. For the signals that we consider in this paper, we can naturally quantify how close two signals are by using the metric d . Let s and

s' be signals in $\mathfrak{F}(R, X)$, then

$$\rho(s, s') = \sup_{t \in R} \{d(s(t), s'(t))\} \quad (3.1)$$

is a metric¹ on the set $\mathfrak{F}(R, X) = X^R$. Since the space of signals is equipped with a metric, we can define a tube around a signal s (see Fig. 3.1). Given an $\varepsilon > 0$, $B_\rho(s, \varepsilon) \subseteq \mathfrak{F}(R, X)$ is the set of all signals that remain ε -close to s .

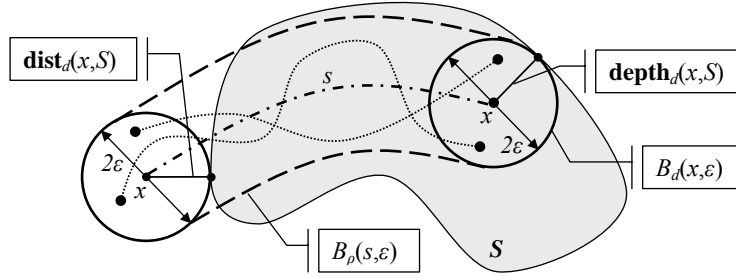


Figure 3.1: The definition of distance and depth and the associated neighborhoods. Also, a tube (*dashed lines*) around a nominal signal s (*dash-dotted line*). The tube encloses a set of signals (*dotted lines*).

Informally, we define the robustness degree to be the bound on the perturbation that the signal can tolerate without changing the truth value of a specification expressed in the Linear [56] or Metric Temporal Logic [116]. Abstractly speaking, the degree of robustness that a signal s satisfies an MTL formula ϕ is a number $\varepsilon \in \overline{\mathbb{R}}$. Intuitively, a positive ε means that the formula ϕ is satisfiable in the Boolean sense and, moreover, that all the other signals that remain ε -close to the nominal one also satisfy ϕ . Accordingly, if ε is negative, then s does not satisfy ϕ and all the other signals that remain within the open tube of radius $|\varepsilon|$ also do not satisfy ϕ .

Definition 3.1.1 (Continuous-Time Robustness Degree). *Let $\phi \in MTL_{\mathbb{B}}(AP)$ be*

¹This is the standard metric - namely the *sup metric* - used in spaces of bounded functions [146, §43]. Since in our definitions we allow a metric to take the value $+\infty$, ρ is also a metric over the set $\mathfrak{F}(R, X)$.

an MTL formula, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ be an observation map and $s \in \mathfrak{F}(R, X)$ be a continuous-time signal, then $\mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O}))$ is the robustness degree of s with respect to ϕ at time t and $\mathbf{Dist}_\rho(s, \mathcal{L}(\phi, \mathcal{O}))$ is the robustness degree of s with respect to ϕ .

The following proposition is a direct consequence of the definitions. It states that all the signals s' , which have distance from s less than the robustness degree of s with respect to ϕ at time t , satisfy the same specification ϕ as s at time t .

Proposition 3.1.1. *Let $\phi \in MTL_{\mathbb{B}}(AP)$, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and $s \in \mathfrak{F}(R, X)$. If $\varepsilon = \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O})) \neq 0$ for some $t \in R$, then for all $s' \in B_\rho(s, |\varepsilon|)$, we have $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s', t) = \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t)$.*

In the following, given an $\varepsilon > 0$ any ball $B_\rho(s, \varepsilon)$ such that for all $s' \in B_\rho(s, \varepsilon)$ we have $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s', t) = \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t)$ will be referred to as *robust neighborhood*. Note that the robustness degree of s with respect to ϕ is actually the radius of the largest robustness neighborhood around s .

Remark 3.1.1. *If $\varepsilon = 0$, then the truth value of ϕ with respect to s is not robust, i.e., there exists some time t such that a small perturbation of the signal's value $s(t)$ can change the Boolean truth value of the formula with respect to s .*

Proposition 3.1.1 has an important implication. If there is a way to guarantee that a set of signals remains δ -close to a signal that satisfies the specification with robustness degree $\varepsilon \geq \delta$, then we can infer that all the other signals in the set also satisfy the same specification. Similarly, if the nominal signal does not satisfy the MTL formula, then no other signal in its ε -neighborhood does. Nevertheless, the set $\mathcal{L}(\phi, \mathcal{O})$ cannot be computed or represented analytically. In the rest of this chapter and in Chapter 5, we develop a series of approximations that will enable us

to compute an under-approximation of the robustness degree by directly operating on a given signal.

3.1.2 Robustness Estimate for Continuous-Time Signals

As explained in the previous section, the robustness degree is the maximum radius of the neighborhood that we can fit around a given signal s without changing the truth value of the formula. But are there other ways to determine and compute robust neighborhoods? In this section, we answer this question in a positive manner by introducing *robust semantics* for MTL formulas.

The robust semantics for MTL formulas are multi-valued semantics over the linearly ordered set $\overline{\mathbb{R}}$. We define the valuation function on the atomic propositions to be the depth (or the distance) of the current value of the signal $s(t)$ in (from) the set $\mathcal{O}(p)$ labeled by the atomic proposition p . Intuitively, this distance represents how robustly is the point $s(t)$ within a set $\mathcal{O}(p)$. If this metric is zero, then even the smallest perturbation of the point can drive it inside or outside the set $\mathcal{O}(p)$ dramatically affecting membership. For the purposes of the following discussion, we use the notation $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)$ to denote the robust valuation of the formula ϕ over the signal s at time t . Formally, $\llbracket \cdot, \cdot \rrbracket_C : (MTL_{\overline{\mathbb{R}} \cup \mathbb{B}}(AP) \times \mathfrak{F}(AP, \mathcal{P}(X))) \rightarrow (\mathfrak{F}(R, X) \times R \rightarrow \overline{\mathbb{R}})$.

Definition 3.1.2 (Continuous-Time Robust Semantics). *Let $s \in \mathfrak{F}(R, X)$, $c \in \overline{\mathbb{R}}$ and $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, then the continuous-time robust semantics of any formula $\phi \in MTL_{\overline{\mathbb{R}} \cup \mathbb{B}}(AP)$ with respect to s is recursively defined as follows*

$$\begin{aligned} \llbracket \top, \mathcal{O} \rrbracket_C(s, t) &:= +\infty \\ \llbracket c, \mathcal{O} \rrbracket_C(s, t) &:= c \\ \llbracket p, \mathcal{O} \rrbracket_C(s, t) &:= \mathbf{Dist}_d(s(t), \mathcal{O}(p)) \end{aligned}$$

$$\llbracket \neg\phi_1, \mathcal{O} \rrbracket_C(s, t) := -\llbracket \phi_1, \mathcal{O} \rrbracket_C(s, t)$$

$$\llbracket \phi_1 \vee \phi_2, \mathcal{O} \rrbracket_C(s, t) := \llbracket \phi_1, \mathcal{O} \rrbracket_C(s, t) \sqcup \llbracket \phi_2, \mathcal{O} \rrbracket_C(s, t)$$

$$\llbracket \phi_1 \mathcal{U}_I \phi_2, \mathcal{O} \rrbracket_C(s, t) := \bigsqcup_{t' \in (t+R\mathcal{I})} (\llbracket \phi_2, \mathcal{O} \rrbracket_C(s, t') \sqcap \prod_{t < t'' < t'} \llbracket \phi_1, \mathcal{O} \rrbracket_C(s, t''))$$

where $t, t', t'' \in R$.

It is easy to verify that the semantics of the negation operator give us all the usual nice properties such as the *De Morgan laws*: $a \sqcup b = -(-a \sqcap -b)$ and $a \sqcap b = -(-a \sqcup -b)$, *involution*: $-(-a) = a$ and *antisymmetry*: $a \leq b$ iff $-a \geq -b$ for $a, b \in \overline{\mathbb{R}}$. Therefore, we can convert any MTL formula ϕ into negation normal form.

Lemma 3.1.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B} \cup \overline{\mathbb{R}}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a continuous-time signal $s \in \mathfrak{F}(R, X)$ and any time $t \in R$, we have $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \llbracket \mathbf{nnf}(\phi), \mathcal{O} \rrbracket_C(s, t)$.*

The next theorem comprises the basic step for establishing that the robust interpretation of an MTL formula ϕ over a signal s evaluates to the radius of a robust neighborhood.

Theorem 3.1.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a continuous-time signal $s \in \mathfrak{F}(R, X)$, then for any $t \in R$, we have $-\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O})) \leq \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O}))$.*

Essentially, Theorem 3.1.1 states that the evaluation of the robust semantics of a formula can be bounded by its robustness degree. In detail, we have : (i) if $s \in \mathcal{L}_t(\phi, \mathcal{O})$, then $0 \leq \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg\phi, \mathcal{O}))$, and if $s \in \mathcal{L}_t(\neg\phi, \mathcal{O})$, then $-\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O})) \leq \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq 0$. Hence, the inequality

$$|\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)| \leq |\mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O}))| \quad (3.2)$$

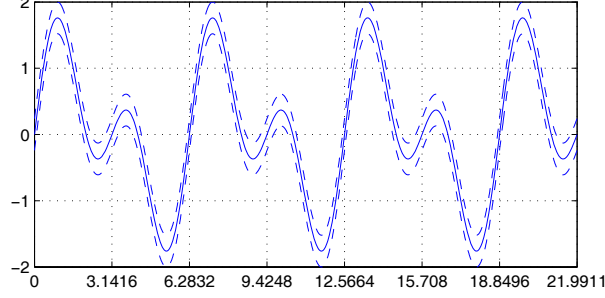


Figure 3.2: The signal s_1 and its robustness neighborhood with radius 0.2398.

holds. Therefore from Theorem 3.1.1 and Proposition 3.1.1, we establish as corollary that the robust interpretation of a formula indeed evaluates to the radius of a robust neighborhood.

Corollary 3.1.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a continuous-time signal $s \in \mathfrak{F}(R, X)$, if for some $t \in R$, we have $\varepsilon = \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \neq 0$, then for all $s' \in B_\rho(s, |\varepsilon|)$, we have $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s', t) = \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t)$.*

Example 3.1.1. *Consider again the continuous-time signal s_1 in Fig. 2.1 and assume that we are given the MTL specification $\phi_0 = \Box p_1 \wedge \Diamond_{[\tau\pi, +\infty)} p_2$, where $\mathcal{O}(p_1) = [-2, 2]$ and $\mathcal{O}(p_2) = (-\infty, 0]$. Then, $\llbracket \phi_0, \mathcal{O} \rrbracket_C(s_1) \approx 0.2398$. Note that any other signal that remains within the tube around s_1 in Fig. 3.2 also satisfies the specification ϕ_0 .*

The following proposition states the relationship between the Boolean and the robust semantics of MTL.

Proposition 3.1.2. *For a formula $\phi \in MTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a continuous-time signal $s \in \mathfrak{F}(R, X)$ and some time $t \in R$, the following results hold*

1. $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) > 0 \Rightarrow \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \top$
2. $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) < 0 \Rightarrow \langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \perp$

$$3. \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \top \Rightarrow \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \geq 0$$

$$4. \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \perp \Rightarrow \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq 0$$

Note that the equivalence in the above proposition fails because, if a point is on the boundary of the set, its distance to the set or its depth in the set is by definition zero. Therefore, the point is classified to belong to that set even if the set is open in the topology.

The previous result raises one important question. If we had a procedure that computes $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)$, can we determine a lower bound on $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)$? In other words, given an $\varepsilon > 0$, can we establish that the robust semantics of ϕ with respect to s evaluate at least to ε ? We can provide a positive answer to this question by simply modifying the observation map \mathcal{O} .

But, first, we must be able to distinguish between negative and positive atomic propositions. That is, atomic propositions with and without a negation operator preceding them. Hence, given an MTL formula ϕ , we consider its equivalent negation normal form $\mathbf{nnf}(\phi)$. However, there might still exist negation operators in the formula $\mathbf{nnf}(\phi)$, namely in front of atomic propositions. Therefore, we introduce an *extended set of atomic propositions* Ξ_{AP} . In detail, we first define two new sets of symbols $\Xi_{AP}^+ = \{\xi_p \mid p \in AP\}$ and $\Xi_{AP}^- = \{\xi_{\neg p} \mid p \in AP\}$ and, then, we set $\Xi_{AP} = \Xi_{AP}^+ \cup \Xi_{AP}^-$. We also define a translation operator $\mathbf{pos} : MTL_{\mathbb{B}}^+(AP) \rightarrow MTL_{\mathbb{B}}^+(\Xi_{AP})$ which takes as input an MTL formula ϕ in NNF and it returns a formula $\mathbf{pos}(\phi)$ where the occurrences of the terms p and $\neg p$ have been replaced by the members ξ_p and $\xi_{\neg p}$ of Ξ_{AP} respectively.

Since we have a new set of atomic propositions, namely Ξ_{AP} , we need to define a new map $\mathcal{O}^e : \Xi_{AP} \rightarrow \mathcal{P}(X)$ for the interpretation of the propositions. This is

straightforward

$$\forall \zeta \in \Xi_{AP}, \mathcal{O}^e(\zeta) =: \begin{cases} \mathcal{O}(p) & \text{if } \zeta = \xi_p \text{ for } p \in AP \\ X \setminus \mathcal{O}(p) & \text{if } \zeta = \xi_{-p} \text{ for } p \in AP \end{cases}$$

Then, the following result is immediate from Lemma 3.1.1 and the definition of \mathcal{O}^e .

Lemma 3.1.2. *Given a signal $s \in \mathfrak{F}(R, X)$, a time $t \in R$, a formula $\phi \in MTL_{\mathbb{B}}(AP)$ and a map $\mathcal{O} : AP \rightarrow \mathcal{P}(X)$, then we have*

1. $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \langle\langle \mathbf{pos}(\mathbf{nnf}(\phi)), \mathcal{O}^e \rangle\rangle_C(s, t)$, and,
2. $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \llbracket \mathbf{pos}(\mathbf{nnf}(\phi)), \mathcal{O}^e \rrbracket_C(s, t)$.

At this point, we have distinguished between the regions that must be avoided (Ξ_{AP}^-) and the regions that must be reached (Ξ_{AP}^+). We proceed to formally define what we mean by region contraction and expansion in order to define the modified (or ε -robustified) map $\mathcal{O}_\varepsilon^e$.

Definition 3.1.3 (ε -Contraction, ε -Expansion). *Let $S \subseteq X$ and $\varepsilon > 0$, then*

- $C_d(S, \varepsilon) = \{x \in X \mid \overline{B_d(x, \varepsilon)} \subseteq S\}$ is the δ -contraction of the set S , and
- $E_d(S, \varepsilon) = \{x \in X \mid \overline{B_d(x, \varepsilon)} \cap S \neq \emptyset\}$ is the δ -expansion of the set S .

Now, the ε -robust interpretation of a given MTL formula ϕ can be achieved by simply introducing a new map $\mathcal{O}_\varepsilon^e : \Xi_{AP} \rightarrow \mathcal{P}(Z)$, where $Z = C_d(X, \varepsilon)$. For a given $\varepsilon \in \mathbb{R}_{\geq 0}$, the definition of the map $\mathcal{O}_\varepsilon^e$ is founded on the map \mathcal{O}^e as follows:

$$\forall \xi \in \Xi_{AP}, \quad \mathcal{O}_\varepsilon^e(\xi) =: \overline{C_d(\mathcal{O}^e(\xi), \varepsilon)}.$$

The following theorem is the connecting link between the robust and Boolean semantics. Informally, it states that given $\varepsilon > 0$, if a continuous-time signal s satisfies the

ε -robust interpretation of the input specification $\phi' = \mathbf{pos}(\mathbf{nnf}(\phi))$, then its robustness with respect to ϕ should be greater than ε .

Theorem 3.1.2. *Let the formula $\phi \in MTL_{\mathbb{B}}(AP)$, the map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, the continuous-time signal $s \in \mathfrak{F}(R, X)$ and the number $\varepsilon \in \mathbb{R}_{>0}$ be given. Set $\phi' = \mathbf{pos}(\mathbf{nnf}(\phi))$, then $\langle\langle \phi', \mathcal{O}_{\varepsilon}^e \rangle\rangle_C(s, t) = \top$ implies $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \geq \varepsilon$.*

At this point, we have not yet answered what is the exact relationship between $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)$ and $\mathbf{Dist}_{\rho}(s, \mathcal{L}_t(\phi, \mathcal{O}))$. For example, could we have replaced the inequality in equation (3.2) with an equality? As the following example indicates, the inequality in equation (3.2) is usually strict. Therefore, in the following we refer to the evaluation of the robust semantics $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t)$ as the *robustness estimate*.

Example 3.1.2. *Consider the constant signal $s(t) = 0$ for $t \geq 0$ and the formula $\psi = \square(p_1 \vee p_2)$ with $\mathcal{O}(p_1) = (-1, 2)$ and $\mathcal{O}(p_2) = (-2, 1)$. It is easy to see that $\mathcal{L}(\psi, \mathcal{O}) = (-2, 2)^{\mathbb{R}_{\geq 0}}$ and, thus, $\mathbf{Dist}_{\rho}(s, \mathcal{L}(\psi, \mathcal{O})) = 2$. However,*

$$\llbracket \psi, \mathcal{O} \rrbracket_C(s) = \prod_{t \geq 0} (\llbracket p_1, \mathcal{O} \rrbracket_C(s, t) \sqcup \llbracket p_2, \mathcal{O} \rrbracket_C(s, t)) = \prod_{t \geq 0} (1 \sqcup 1) = 1.$$

In other words, the robust MTL semantics evaluate to an under-approximation of the robustness degree.

Unfortunately, the robust semantics cannot always capture the fact that a signal is robust with respect to a specification. The next example demonstrates how the robustness estimate might evaluate to zero even when the formula is valid.

Example 3.1.3. *Consider the constant signal $s(t) = 0$ for $t \geq 0$ and the formula $\psi = \square(p_1 \vee p_2)$ with $\mathcal{O}(p_1) = [0, +\infty)$ and $\mathcal{O}(p_2) = (-\infty, 0]$. Clearly, $\mathcal{L}(\psi, \mathcal{O}) = \mathbb{R}^{\mathbb{R}_{\geq 0}}$, i.e., the formula is valid, and, thus, $\mathbf{Dist}_{\rho}(s, \mathcal{L}(\psi, \mathcal{O})) = +\infty$. However, $\llbracket \psi, \mathcal{O} \rrbracket_C(s) = \prod_{t \geq 0} (\llbracket p_1, \mathcal{O} \rrbracket_C(s, t) \sqcup \llbracket p_2, \mathcal{O} \rrbracket_C(s, t)) = \prod_{t \geq 0} (0 \sqcup 0) = 0$.*

These undesirable effects can be minimized or even avoided if we understand how equality breaks. Generally speaking, the strict inequality between the robustness estimate and robustness degree manifests itself in four distinct ways: (i) at the level of the atomic propositions, e.g., $p_1 \vee p_2$, (ii) due to the existence of tautologies in the formula, e.g., $p \vee \neg p$, (iii) when we consider disjuncts of MTL subformulas, e.g., $\phi_1 \vee \phi_2$, and more importantly, (iv) due to the supremum operator in the semantics of the until temporal operator. The mathematical principle that is behind the above four cases is the fact that the distance of a point from the intersection of two sets is not equal to the maximum of the distance of the point from each set [139]. The details of how this is manifested in the relationship between the robustness degree and the robustness estimate can be found in the proof of Theorem 3.1.1 in Appendix 8.1.

The first case can be remedied by introducing a new symbol for each Boolean combination of atomic propositions. The second and third conditions require the attention of the user of the algorithm. Even though the above cases can be fixed by introducing syntactic restrictions, the last case captures the fundamental difference between the robustness degree and the robustness estimate. The signals in $B_\rho(s, |\varepsilon|)$ with $\varepsilon = \mathbf{Dist}_\rho(s, \mathcal{L}(\phi, \mathcal{O}))$ can satisfy or falsify the specification ϕ at different time instants than s . On the other hand, the robustness estimate returns the radius of the robust neighborhood of signals that satisfy the specification at the same point in time.

However, some applications (see for example [59] or Section 6) might benefit from specifications ϕ in MTL for which the equality holds, that is,

$$\llbracket \phi, \mathcal{O} \rrbracket_C(s) = \mathbf{Dist}_\rho(s, \mathcal{L}(\phi, \mathcal{O})). \quad (3.3)$$

The following result indicates a fragment of MTL for which equality (3.3) holds. Note that we have not imposed any conditions on the metric d , the set $\mathfrak{F}(R, X)$ or the topology of the sets $\mathcal{O}(p)$.

Proposition 3.1.3. *Consider a formula $\phi \in MTL_{\mathbb{B}}^+(AP, \wedge, \square)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a signal $s \in \mathfrak{F}(R, X)$, then for any $t \in R$, $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \top$ implies $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O}))$.*

Since duality holds in our definition of the robust semantics, the next result is immediate.

Corollary 3.1.2. *Consider a formula $\phi \in MTL_{\mathbb{B}}^+(AP, \vee, \diamond)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a signal $s \in \mathfrak{F}(R, X)$, then for any $t \in R$, $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(s, t) = \perp$ implies $\llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi, \mathcal{O}))$.*

Remark 3.1.2. *Further results on the equality (3.3) are possible on a case-by-case basis by imposing additional conditions on the metric d , the sets $\mathfrak{F}(R, X)$ and X (and their respective topologies) and/or the structure of the sets $\mathcal{O}(p)$.*

Remark 3.1.3. *The results in Section 2 hold for any linearly ordered time domain and not just the real line. As it can be seen in the proofs in Appendix 8.1, the only requirement is that the timing constraints \mathcal{I} in a formula ϕ must refer to the same time domain as the domain of the signal s . For example, consider a discrete-time signal $\sigma \in \mathfrak{F}(N, X)$, where $N \subseteq \mathbb{N}$, and the formula $\diamond_{[1,3]}p_1$. In this case, the timing constraints $[1, 3]$ refer to the discrete-time domain of σ where the time now counts clock ticks or samples instead of real time.*

3.2 Discrete Time

3.2.1 Robustness Degree for Timed State Sequences

Similarly to the continuous-time case, we define a metric for the discrete-time signals.

Let σ and σ' be discrete-time signals in $\mathfrak{F}(N, X)$, then

$$\hat{\rho}(\sigma, \sigma') = \sup_{i \in N} \{d(\sigma(i), \sigma'(i))\} \quad (3.4)$$

is a metric on the set $\mathfrak{F}(N, X) = X^N$. Now, the neighborhood $B_{\hat{\rho}}(\sigma, \varepsilon)$ contains all the discrete-time signals that remain ε -close to σ . For notational convenience, we extend the definition of the neighborhood B over timed state sequences under the same timing function τ as follows. If $\mu = (\sigma, \tau)$, $\mu' = (\sigma', \tau)$ and $s' \in B_{\hat{\rho}}(\sigma, \varepsilon)$, then $\mu' \in B_{\hat{\rho}}(\mu, \varepsilon)$. The formulation of the robustness degree for the discrete-time case is straightforward.

Definition 3.2.1 (Discrete-Time Robustness Degree). *Consider an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^{\dagger}(N, \mathbb{R}_{\geq 0})$, then $\mathbf{Dist}_{\hat{\rho}}(\sigma, \mathcal{L}_i^{\tau}(\phi, \mathcal{O}))$, where $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$, is the discrete-time robustness degree of μ with respect to ϕ at time $i \in N$ and $\mathbf{Dist}_{\hat{\rho}}(\sigma, \mathcal{L}^{\tau}(\phi, \mathcal{O}))$ is the discrete-time robustness degree of μ with respect to ϕ .*

As before, the following proposition is derived directly from the definitions. It states that all the timed state sequences μ' which have the same timing function τ with μ and whose discrete-time signals $\mu'^{(1)}$ have distance from $\mu^{(1)}$ less than the robustness degree of μ with respect to ϕ satisfy the same specification ϕ as μ under τ .

Proposition 3.2.1. *Let $\phi \in MTL_{\mathbb{B}}(AP)$ be an MTL formula, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ be an observation map and $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^{\dagger}(N, \mathbb{R}_{\geq 0})$ be a timed state sequence. Also,*

let $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$. If $\varepsilon = \mathbf{Dist}_{\hat{\rho}}(\sigma, \mathcal{L}_i^\tau(\phi, \mathcal{O})) \neq 0$ for some $i \in N$, then for all $\mu' = (\sigma', \tau)$ such that $\sigma' \in B_{\hat{\rho}}(\sigma, |\varepsilon|)$, we have $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu', i) = \langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i)$.

A major advantage of discrete-time robustness, when compared to the continuous-time case, is that now the set $\mathcal{L}^\tau(\phi, \mathcal{O})$ can be computed when $N = \mathbf{dom}(\tau)$ is a finite set. In [150], it was proven the one can construct an acceptor \mathcal{A}_ϕ (in the form of a timed alternating automaton with one clock) for the finite models of any formula ϕ in the logic MTL with point-based semantics. Assume now that we are given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$ and a timing function $\tau \in \mathfrak{F}^\uparrow(N, \mathbb{R}_{\geq 0})$. For that particular τ , we can find the set $TSS^\tau(\mathcal{A}_\phi)$ of all timed state sequences (or timed words) (w, τ) with $w \in \mathcal{F}(N, \mathcal{P}(AP))$ that are accepted by \mathcal{A}_ϕ . One way to do so is to construct the set W of all possible untimed words w of length $|N|$, that is $W = \mathcal{F}(N, \mathcal{P}(AP))$, and, then, for each $w \in W$ verify whether (w, τ) is accepted by \mathcal{A}_ϕ , i.e., whether $(w, \tau) \in TSS^\tau(\mathcal{A}_\phi)$. From the set $TSS^\tau(\mathcal{A}_\phi)$, we can easily derive the set

$$\mathcal{L}^\tau(\phi, \mathcal{O}) = \bigcup_{(w, \tau) \in TSS^\tau(\mathcal{A}_\phi)} \prod_{i \in N} \left(\bigcap_{p \in w(i)} \mathcal{O}(p) \cap \bigcap_{p \in AP \setminus w(i)} X \setminus \mathcal{O}(p) \right).$$

The following example illustrates the concept of robustness for temporal logic formulas interpreted over finite (timed) state sequences.

Example 3.2.1. Assume that we are given the LTL specification $\phi = p_1 \mathcal{U} p_2$ such that $\mathcal{O}(p_1) = [1, 2] \subseteq \mathbb{R}$ and $\mathcal{O}(p_2) = [0, 1] \subseteq \mathbb{R}$. Note that the sets $\mathcal{O}(p_1)$ and $\mathcal{O}(p_2)$ are disjoint. Consider now two timed state sequences $\mu_1 = (\sigma_1, \tau)$ and $\mu_2 = (\sigma_2, \tau)$ with time domain $N = \{0, 1\}$ taking values in \mathbb{R} such that $\sigma_1(0) = 1$, $\sigma_1(1) = 0.5$ and $\sigma_2(0) = 1.7$, $\sigma_2(1) = 1.3$. In this simple case, we can compute the set $\mathcal{L}^\tau(\phi, \mathcal{O})$ with the procedure described above. The four untimed words that generate non-empty sets and satisfy the specification ϕ are $w_1 = (\{p_2\}, \{p_1\})$, $w_2 = (\{p_2\}, \{p_2\})$, $w_3 = (\{p_2\}, \emptyset)$

and $w_4 = (\{p_1\}, \{p_2\})$. Hence, we get $\mathcal{L}^\tau(\phi, \mathcal{O}) = \mathcal{O}(p_2) \times \mathcal{O}(p_1) \cup \mathcal{O}(p_2) \times \mathcal{O}(p_2) \cup \mathcal{O}(p_2) \times X \setminus (\mathcal{O}(p_1) \cup \mathcal{O}(p_2)) \cup \mathcal{O}(p_1) \times \mathcal{O}(p_2) = [0, 1) \times \mathbb{R} \cup [1, 2] \times [0, 1)$ (see Fig. 3.3).

Therefore, $\varepsilon_1 = \mathbf{Dist}_{\hat{\rho}}(\sigma_1, \mathcal{L}^\tau(\phi, \mathcal{O})) = 0.5$ and $\varepsilon_2 = \mathbf{Dist}_{\hat{\rho}}(\sigma_2, \mathcal{L}^\tau(\phi, \mathcal{O})) = -0.3$.

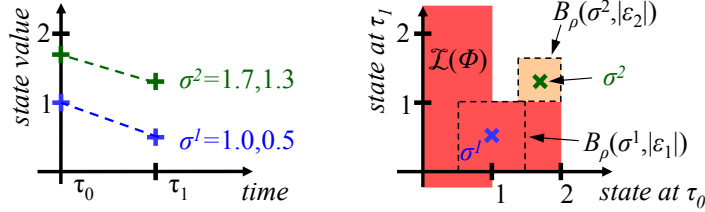


Figure 3.3: On the left appears the time-domain representation of the discrete-time signals σ_1 and σ_2 of Example 3.2.1. On the right appears the space of the discrete-time signals of length 2. Each x represents a signal as a point in \mathbb{R}^2 .

Remark 3.2.1. *In the case of LTL, the construction of the set $\mathcal{L}^\tau(\phi, \mathcal{O})$ can be slightly improved. Giannakopoulou and Havelund [75] have developed an efficient algorithm for the translation of LTL formulas over finite traces to finite automata.*

3.2.2 Robustness Estimate for Timed State Sequences

The aforementioned theoretical construction of the set $\mathcal{L}^\tau(\phi, \mathcal{O})$ cannot be of significant practical interest. Moreover, the definition of robustness degree involves a number of set operations (union, intersection and complementation) in the possibly high dimensional spaces X and $\mathfrak{F}(N, X)$, which can be computationally expensive in practice. Fortunately, the discrete-time robust semantics of MTL can provide us with a feasible method for under-approximating the robustness degree of (finite) timed state sequences.

Definition 3.2.2 (Discrete-Time Robust Semantics). *Let $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, $c \in \overline{\mathbb{R}}$ and $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, then the discrete-time robust semantics of any formula*

$\phi \in MTL_{\mathbb{R} \cup \mathbb{B}}(AP)$ with respect to μ is recursively defined as follows

$$\begin{aligned}
\llbracket \top, \mathcal{O} \rrbracket_D(\mu, i) &:= +\infty \\
\llbracket c, \mathcal{O} \rrbracket_D(\mu, i) &:= c \\
\llbracket p, \mathcal{O} \rrbracket_D(\mu, i) &:= \mathbf{Dist}_d(\sigma(i), \mathcal{O}(p)) \\
\llbracket \neg\phi_1, \mathcal{O} \rrbracket_D(\mu, i) &:= -\llbracket \phi_1, \mathcal{O} \rrbracket_D(\mu, i) \\
\llbracket \phi_1 \vee \phi_2, \mathcal{O} \rrbracket_D(\mu, i) &:= \llbracket \phi_1, \mathcal{O} \rrbracket_D(\mu, i) \sqcup \llbracket \phi_2, \mathcal{O} \rrbracket_D(\mu, i) \\
\llbracket \phi_1 \mathcal{U}_I \phi_2, \mathcal{O} \rrbracket_D(\mu, i) &:= \bigsqcup_{j \in \tau^{-1}(\tau(i)+I)} (\llbracket \phi_2, \mathcal{O} \rrbracket_D(\mu, j) \sqcap \prod_{i \leq k < j} \llbracket \phi_1, \mathcal{O} \rrbracket_D(\mu, k))
\end{aligned}$$

where $i, j, k \in N$, $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$.

Similarly to the continuous-time robust semantics, the following results hold.

Lemma 3.2.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$ and any time $i \in N$, we have $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) = \llbracket \mathbf{nnf}(\phi), \mathcal{O} \rrbracket_D(\mu, i)$.*

Again, the robust semantics evaluate to the radius of a robust neighborhood.

Theorem 3.2.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a timed state sequence $\mu = (\sigma, \tau) \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, then for any $i \in N$, we have $-\mathbf{dist}_{\hat{\rho}}(\sigma, \mathcal{L}_i^\tau(\phi, \mathcal{O})) \leq \llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) \leq \mathbf{depth}_{\hat{\rho}}(\sigma, \mathcal{L}_i^\tau(\phi, \mathcal{O}))$.*

That is, the inequality $|\llbracket \phi, \mathcal{O} \rrbracket_D(\mu)| \leq |\mathbf{Dist}_{\hat{\rho}}(\mu^{(1)}, \mathcal{L}^{\mu^{(2)}}(\phi, \mathcal{O}))|$ holds in the case of discrete-time semantics, too. In addition, we get the following Corollary.

Corollary 3.2.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, let $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$. If for some $i \in N$ we have $\varepsilon = \llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) \neq 0$, then for all $\mu' = (\sigma', \tau)$ such that $\sigma' \in B_{\hat{\rho}}(\sigma, |\varepsilon|)$ we have $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu', i) = \llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i)$.*

Moreover, the relationship between robust and Boolean semantics in discrete-time is maintained.

Proposition 3.2.2. *For an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^{\uparrow}(N, \mathbb{R}_{\geq 0})$ and some time instant $i \in N$, the following results hold*

1. $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) > 0 \Rightarrow \langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \top$
2. $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) < 0 \Rightarrow \langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \perp$
3. $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \top \Rightarrow \llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) \geq 0$
4. $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \perp \Rightarrow \llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) \leq 0$

Finally, we close this section by restating Proposition 3.1.3 and Corollary 3.1.2 for discrete-time semantics.

Proposition 3.2.3. *Consider a formula $\phi \in MTL_{\mathbb{B}}^+(AP, \wedge, \square)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^{\uparrow}(N, \mathbb{R}_{\geq 0})$, then for any $i \in N$, $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \top$ implies $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) = \mathbf{Dist}_{\hat{\rho}}(\sigma, \mathcal{L}_i^{\tau}(\phi, \mathcal{O}))$, where $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$.*

Corollary 3.2.2. *Consider a formula $\phi \in MTL_{\mathbb{B}}^+(AP, \vee, \diamond)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^{\uparrow}(N, \mathbb{R}_{\geq 0})$, then for any $i \in N$, $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \perp$ implies $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) = \mathbf{Dist}_{\hat{\rho}}(\sigma, \mathcal{L}_i^{\tau}(\phi, \mathcal{O}))$, where $\sigma = \mu^{(1)}$ and $\tau = \mu^{(2)}$.*

3.2.3 Testing the Robustness of Temporal Properties

In this section, we present a procedure that computes the robustness estimate of a finite timed state sequence μ with respect to a specification ϕ stated in the Metric

Temporal Logic. For this purpose, we design a monitoring algorithm based on the robust semantics of MTL.

Similarly to the monitoring algorithm in [174], we start from the definition of the robust semantics of the until operator and using the distributive law (see Appendix 8.2.1), we can derive an equivalent formulation (we have omitted the map \mathcal{O}):

$$\llbracket \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rrbracket_D(\mu, i) = \begin{cases} (K_{\varepsilon}^{\infty}(0, \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i)) \sqcup \\ \sqcup (\llbracket \phi_1 \rrbracket_D(\mu, i) \sqcap \llbracket \phi_1 \mathcal{U}_{\mathcal{I}-R\delta\tau(i)} \phi_2 \rrbracket_D(\mu, i+1)) & \text{if } i < \max N \\ K_{\varepsilon}^{\infty}(0, \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i) & \text{otherwise} \end{cases}$$

where $\tau = \mu^{(2)}$, $N = \mathbf{dom}(\tau)$, $\delta\tau(i) = \tau(i+1) - \tau(i)$ and $K_{\varepsilon}^{\infty}(a, A) = +\infty$ if $a \in A$ and $-\infty$ otherwise.

Algorithm 1 Monitoring the Robustness of Timed State Sequences

Input: An MTL formula ϕ , a finite timed state sequence $\mu = (\sigma, \tau)$ and a predicate map \mathcal{O}

Output: The formula's robustness estimate

```

1: procedure MONITOR( $\phi, \mu, \mathcal{O}$ )
2:    $i \leftarrow 0$ 
3:   while  $\phi \neq \varepsilon \in \overline{\mathbb{R}}$  do  $\triangleright \phi$  has not been reduced to a value
4:     if  $i < \max \mathbf{dom}(\tau)$  then  $\phi \leftarrow \text{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O})$ 
5:     else  $\phi \leftarrow \text{DERIVE}(\phi, \sigma(i), 0, \top, \mathcal{O})$ 
6:     end if
7:      $i \leftarrow i + 1$ 
8:   end while
9: end procedure

```

Using the recursive definition, it is easy to derive Algorithm 1 that returns the robustness estimate of a given finite timed state sequence μ with respect to an MTL formula ϕ . Algorithm 2 is the core of the monitoring procedure. It takes as input the temporal logic formula ϕ , the current state $\sigma(i)$ and the time period before the next state occurs, it evaluates the part of the formula that must hold on the current state and returns the formula that it has to hold at the next state of the timed state

Algorithm 2 Deriving the Future

Input: The MTL formula ϕ , the current value of the signal x , the time period δt before the next value in the signal, a variable $last$ indicating whether the next state is the last and the predicate map \mathcal{O}

Output: The MTL formula ϕ that has to hold at the next moment in time

```
1: procedure DERIVE( $\phi, x, \delta t, last, \mathcal{O}$ )
2:   if  $\phi = \top$  then return  $+\infty$ 
3:   else if  $\phi = \varepsilon \in \overline{\mathbb{R}}$  then return  $\varepsilon$ 
4:   else if  $\phi = p \in AP$  then return  $\text{Dist}_d(x, \mathcal{O}(p))$ 
5:   else if  $\phi = \neg\phi_1$  then return  $\neg\text{DERIVE}(\phi_1, x, \delta t, last, \mathcal{O})$ 
6:   else if  $\phi = \phi_1 \vee \phi_2$  then
7:     return  $\text{DERIVE}(\phi_1, x, \delta t, last, \mathcal{O}) \vee \text{DERIVE}(\phi_2, x, \delta t, last, \mathcal{O})$ 
8:   else if  $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$  then
9:      $\alpha \leftarrow K_{\varepsilon}^{\infty}(0, \mathcal{I}) \wedge \text{DERIVE}(\phi_2, x, \delta t, last, \mathcal{O})$ 
10:    if  $last = \top$  then return  $\alpha$ 
11:    else return  $\alpha \vee (\text{DERIVE}(\phi_1, x, \delta t, last, \mathcal{O}) \wedge \phi_1 \mathcal{U}_{\mathcal{I}-\delta t} \phi_2)$ 
12:    end if
13:  end if
14: end procedure
```

sequence.

In order to avoid the introduction of additional connectives in our logic that would unnecessarily increase the length of this paper, we have presented Algorithm 2 merely as rewriting procedure on the input formula ϕ . This implies that the procedure MONITOR would return a Boolean combination ψ of numbers from $\overline{\mathbb{R}}$. Then, the robustness estimate would simply be $\llbracket \psi, \mathcal{O} \rrbracket_D(\mu)$. For example, if $\psi = \bigwedge_{a \in A} \bigvee_{b \in B_a} c_{ab}$ with $c_{ab} \in \overline{\mathbb{R}}$, then $\llbracket \psi, \mathcal{O} \rrbracket_D(\mu) = \prod_{a \in A} \sqcup_{b \in B_a} c_{ab}$. In an implementation of the algorithm, the following simplifications must be performed at each call of Algorithm 2 : $\varepsilon_1 \vee \varepsilon_2$ is replaced by $\varepsilon = \varepsilon_1 \sqcup \varepsilon_2$, $\neg\varepsilon$ is replaced by $-\varepsilon$ and, also, $\phi \wedge +\infty \equiv \phi$, $\phi \vee -\infty \equiv \phi$, $\phi \vee +\infty \equiv +\infty$ and $\phi \wedge -\infty \equiv -\infty$.

The following lemma is immediate since the formulation of until in Algorithm 2 is equivalent with the robust interpretation of until in Definition 3.2.2.

Lemma 3.2.2. *Given an MTL formula $\phi \in \text{MTL}_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$*

and a finite timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, then for any $i < \max N$ we have $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) = \llbracket \text{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1)$, where $\sigma = \mu^{(1)}$, $\tau = \mu^{(2)}$ and $N = \mathbf{dom}(\tau)$.

Using Lemma 3.2.2 and the fact that the temporal operators are eliminated from ϕ when $last = \top$, we derive the following theorem as corollary.

Theorem 3.2.2. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a finite timed state sequence $\mu \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$, then*

$$\llbracket \phi, \mathcal{O} \rrbracket_D(\mu) = \llbracket \text{MONITOR}(\phi, \mu, \mathcal{O}) \rrbracket_D(\mu).$$

The theoretical complexity of the Boolean-valued monitoring algorithms has been studied in the past for both the Linear [138] and the Metric Temporal Logic [174]. Practical algorithms for monitoring of Boolean-valued finite timed state sequences using rewriting have been developed by several authors [92, 119].

Essentially, the new part in Algorithm 2 - when compared with Boolean monitoring - is the evaluation of the atomic propositions. How easy is to compute the signed distance? When the set X is just \mathbb{R} , the set S is an interval and the metric d is the function $d_1(x, y) = |x - y|$, then the problem reduces to finding the minimum of two values. For example, if $S = [a, b] \subseteq \mathbb{R}$ and $x \in S$, then $\mathbf{Dist}_d(x, S) = \min\{|x - a|, |x - b|\}$. When the set X is \mathbb{R}^n , $S \subseteq \mathbb{R}^n$ is a convex set and the metric d is the Euclidean distance, i.e., $d_e(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$, then we can calculate the distance (\mathbf{dist}_d) by solving very efficient convex optimization problems. If, in addition, the set S is just a halfspace $S = \{x \mid a^T x \leq b\}$, then there exists an analytical solution : $\mathbf{dist}_d(x, S) = |b - a^T x|/\|a\|$ if $a^T x > b$ and 0 if $a^T x \leq b$. Moreover, if the set S is a concave set defined by a finite union of halfspaces S_i , i.e., $S = \cup_{i \in I} S_i$, then the distance of a point x from S is simply

$\mathbf{dist}_d(x, S) = \min_{i \in I} \mathbf{dist}_d(x, S_i)$. Similar results hold for ellipsoidal sets. For further details on such distance computation problems see [26, §8].

The theoretical complexity of Algorithm 1 is an open problem which we plan to address in the future. Note however that the theoretical running times of convex optimization algorithms are only approximate (see Part III in [26]) and, thus, they do not capture the efficient running times of actual practical implementations. Nevertheless, it is immediate that the theoretical complexity of Algorithm 1 cannot be easier than the complexity of the Boolean monitoring algorithms in [138, 174].

3.3 TALIRO

Temporal Logic RObustness (or TALIRO) is a tool that computes the robustness estimate ε of an MTL formula ϕ with respect to a finite timed state sequence μ . Version 0.1 is available in two formats (Windows and Linux) and it supports only 1D signals. When TALIRO is executed without any input arguments, i.e., `taliro`, then it takes as input arguments the demo input files `demo_spec.txt` and `demo_data.txt` that are distributed with the tool. The input arguments to `taliro` must be two input files, e.g., `taliro inputspec.txt inputdata.txt`.



This is a console application. In order to execute TALIRO, you have to open a console window in the MS Windows family products : `start -> run`, then type `cmd` and change to the directory where you have unzipped the software package. Note that in certain versions of Linux systems you might have to type `./taliro` instead of `taliro` in order to run the software.

3.3.1 Explanation of Input Arguments

Usage `taliro inputspec.txt inputdata.txt`

The file `inputspec.txt`, as the name implies, includes the MTL or LTL formula as well as the observation map \mathcal{O} and some auxiliary variables. A typical input in ASCII format is the following.

```
01. % Demo input specification file for TaLiRo
02. % G. Fainekos - GRASP Lab - 2008.01.22
03.
04. [](p1-><>_(0,.5) !p1)
05.
06. signal dimension : 1
07.
08. number of predicates : 3
09.
10. p1 number of constraints : 1
11. -1.0 -1.0
12.
13. p2 number of constraints : 2
14. 1.0 0.5
15. -1.0 0.5
16.
17. p3 number of constraints : 1
18. 1.0 -1.0
19.
20. timing constraints on the number of samples : no
21. number of samples : 3142
```

The lines that start with the special character `%` are comment lines, e.g. lines 01

and 02. The empty lines, e.g., 03, 05, are not required, however they make the text more readable.

Line 04 is the MTL or LTL formula. Table 3.1 indicates the correspondence between the symbols of the logical operators and the input ASCII characters. The


\neg	\vee	\wedge	\rightarrow	\leftrightarrow
!	\bigvee	\bigwedge	\rightarrow	\leftrightarrow
\square	\diamond	\mathcal{U}	\mathcal{R}	
[]	$\langle \rangle$	\mathcal{U}	\mathcal{R}	

Table 3.1: Correspondence between logical operators and ASCII symbols.

timing constraints on the temporal operators follow the temporal operator using an underscore. That is, if $T \in \{ [], \langle \rangle, \mathcal{R}, \mathcal{U} \}$, then T_I is a temporal operator with timing constraints. In turn, the timing constraints I can have the form $\ll a, b \gg$, where $\ll \in \{ (, [\}$, $\gg \in \{),] \}$ and $a, b \in \mathbb{Q} \cup \{\pm\text{inf}\}$. Currently, no negative numbers are allowed in the timing constraints I since we do not consider past operators. Some examples of timing constraints on the temporal operators are :

$$\mathcal{U}_{(0.23, 5.12)}, []_{[0, 30]}, \mathcal{R}_{[10, \text{inf})}, \langle \rangle_{[2, 2]}$$

Finally, if there is no timing constraint after a temporal operator, then $I = [0, +\text{inf})$ is implied².

 If the timing constraints refer to the actual evolution of time, that is, to the timestamps $\tau(i)$ of a timed state sequence $\mu = (\bar{\sigma}, \tau)$, then we have to store the bounds of I using double precision floating point variables. In this case, comparisons that involve equality ($\leq, \geq, =$) become dubious and should be avoided, i.e., avoid using closed $[\cdot, \cdot]$ or half-closed $[\cdot, \cdot)$, $(\cdot, \cdot]$ intervals. On the other hand, if the timing constraints refer to the number of samples, then the bounds on I are stored using integer variables and comparisons involving equalities are meaningful.

²In future versions of TALIRO, if no temporal operator in the formula has timing constraints, then the formula will be tested using a more memory efficient LTL version of the algorithm.

Line 06 refers to the dimension n of the space \mathbb{R}^n that the signal takes values. Version 0.1 of TALiRO only supports 1D signals, i.e., $n = 1$. This line is added for compatibility with future versions.

Line 08 refers to the number of predicates which are in the domain of the observation map \mathcal{O} . The following lines (09-19) contain the definition of each set that is labeled by an atomic proposition. The declaration of each set begins with the statement :

`<predicate> number of constraints : <m>`

In the position of `<predicate>`, we can place any predicate name that is a combination of alphanumeric characters, e.g., `pred1`, `p1`, `bb`, `aa123bb` etc. For computational reasons, the subsets of \mathbb{R}^n are defined using intersections of halfspaces. This implies that the sets labeled by the atomic propositions are actually convex polyhedral sets. Note that this is not a fundamental restriction of the toolbox, since concave sets can be defined by taking the negation of an atomic proposition. The number `m` denotes how many halfspaces define a set. Each halfspace i of the set is represented by an inequality of the form $\sum_{j=1}^n a_{ij}\sigma^{(j)} \leq b_i$, where $\sigma^{(j)}$ is the j -th component (or continuous variable) of signal σ and $a_{ij}, b_i \in \mathbb{R}$. Then, the set $\mathcal{O}(\text{< predicate >})$ can be defined by the conjunction of the aforementioned inequalities : $\bigwedge_{i=1}^m \sum_{j=1}^n a_{ij}\sigma^{(j)} \leq b_i$. The latter can also be represented by a matrix inequality $A\sigma \leq B$, where $A = \{a_{ij}\}$ and $B = \{b_i\}$. The matrices A and B are given as inputs to TALiRO in the form of a concatenated matrix $[A|B]$. For example, consider the atomic proposition `p2` defined in lines 13-15 which has two constraints. The first constraint indicates that $\sigma^{(1)} \leq 0.5$, while the second that $-\sigma^{(1)} \leq 0.5$. In other words, $\mathcal{O}(\text{p2}) = [-0.5, 0.5]$. In lines 10-11, the atomic proposition `p1` defines the set $\mathcal{O}(\text{p1}) = [1, +\infty)$.



Version 0.1 of TALiRO does not check for emptiness of the sets. Future versions

will have this functionality.

Line 20 indicates whether the timing constraints refer to the number of sampling points or not. In cases where the sampling step is constant, i.e., $\tau(i + 1) - \tau(i) = \Delta\tau \in \mathbb{Q}$ for all $i > 0$, then it might be beneficial to write the timing constraints on the temporal operators with respect to the number of sampling points instead the actual time. For example, if $\Delta\tau = 0.1$, then the formula $\langle \rangle_{[0.1, 0.5]} p1$ can be converted to $\langle \rangle_{[1, 5]} p1$. In the latter case, the equality checks become meaningful since we require that $p1$ holds at some point between the next and the next five samples.



If the answer in line 20 is `no` or the temporal logic formula is in LTL, then we must still provide the timestamps in the file `inputdata.txt` even though the timestamps are ignored by the algorithm.

Finally, line 21 indicates the length of the input timed state sequence.

The file `inputdata.txt` contains the timestamps and the data of the discrete-time signal. The first column contains the timestamps of the samples, while the following columns contain the data for each dimension of the signal. The following table presents the first 5 lines of such an input file generated for an 1D signal. In this example, the sampling step is constant with $\Delta\tau = 0.01$.

0.000000e+000	0.000000e+000
1.000000e-002	2.3998800e-002
2.000000e-002	4.7990401e-002
3.000000e-002	7.1967605e-002
4.000000e-002	9.5923223e-002
	⋮

3.3.2 Examples

The examples presented below were run on PIII 1.2GHz with 1GB RAM under Windows XP. First, assume that we are given the discrete-time signal σ_1 (see Fig. 2.2) and the corresponding timing function τ_1 . The signal σ_1 has 110 sampling points. We would like to verify that whenever the value of the signal raises above the value 1.5, then it drops below 1.5 within 1 time unit. This can be formally stated with the MTL formula

$$\square (p1 \rightarrow \langle \rangle_{(0.0, 1.0)} ! p1) \quad (3.5)$$

where $\mathcal{O}(p1) = [1.5, +\infty)$. The output of TALiRO for this case is

```
robustness : 0.097603
total running time : 0.030000 sec
```

In this example, since the sampling step is constant, i.e., 0.2, we can test the same specification over the number of samples and include the upper bound of the timing constraint – if this is desirable. Then, the formula becomes

$$\square (p1 \rightarrow \langle \rangle_{(0, 5]} ! p1) \quad (3.6)$$

The output of TALiRO is

```
robustness : 0.317274
total running time : 0.020000 sec
```

If we reduce the timing constraint to 0.5, i.e., the MTL formula is

$$\square (p1 \rightarrow \langle \rangle_{(0.0, 0.5)} ! p1) \quad (3.7)$$

then the specification does not hold any more (`robustness` : -0.158058). What if we don't only want the signal value to drop below 1.5, but also to stay below 1.5

for at least 2 time units? Then, we can use the MTL formula

$$\square (p1 \rightarrow \langle \rangle_{(0,5)} \square_{[0,10]} !p1) \quad (3.8)$$

where the constraints are on the number of samples. The result is

```
robustness : 0.097603
total running time : 0.140000 sec
```

Now, if we increase the *always* bounds from 2 time units to 10, that is,

$$\square (p1 \rightarrow \langle \rangle_{(0.0,1.0)} \square_{(0.0,10.0)} !p1) \quad (3.9)$$

where the constraints are on the actual time, then the specification does not hold (**robustness** : -0.250768). Next, assume that we would like to test whether the signal oscillates between the sets $p2$ and $p1$, where $\mathcal{O}(p2) = (-\infty, -1.5]$, in that order. The LTL formula

$$\square (\langle \rangle (p2 \wedge \langle \rangle p1)) \quad (3.10)$$

does not do the job. Even though the signal is periodic and event $p1$ follows $p2$, the robustness of the formula is -1.683066. Here, by p event we mean that the value of the signal is within the set $\mathcal{O}(p)$. This situation occurs because the signal is of finite duration and the *always* operator in formula 3.10 requires that that the subformula $\langle \rangle (p2 \wedge \langle \rangle p1)$ holds at the last sample of the signal (which is obviously not true since there are no more sampling points). If the period of the signal can be estimated, then we could use that as an upper bound on the *always* operator. For example, for signal σ_1 the period is 2π so we could instead test the formula

$$\square_{[0.0,12.57]} (\langle \rangle (p2 \wedge \langle \rangle p1)) \quad (3.11)$$

which is correct with robustness 0.238435. This example implies that we should sample or simulate the discrete-time signal for some time that is longer than the time interval that we would like to test for periodicity. In addition, we can add constraints on the occurrence of the events. For example, for the input formula

$$\square_{[0.0, 12.57]} (\langle \rangle_{[0.0, 6.28]} (p2 / \langle \rangle_{[0.0, 3.14]} p1)) \quad (3.12)$$

we get

```
robustness : 0.238435
total running time : 0.781000 sec
```

Note that formulas (3.10), (3.11) and (3.12) ignore (i) the fact that initially the event $p1$ happens before the event $p2$ and (ii) that the signal σ_1 can take values inside and outside the set $\mathcal{O}(p2)$ several times before event $p1$ is observed.

Tables 3.2 and 3.3 indicate how the computation time of TALIRO changes as the length of the discrete-time signal increases. The sampling function is again τ_1 . For these experimental data, the computer used was a Dell PowerEdge 1650 with two 1.4GHz Pentium-III CPUs and 2GB of RAM running SUSE 9.2 Linux. We can see that the computation time and the memory requirements do not only depend on the length of the discrete-time signal, but also on the structure of the formula. A heuristic way to reduce the computation time and the memory requirements of TALIRO is to add timing constraints on the temporal operators whenever this is possible. Notice that in Tables 3.2 and 3.3 the robustness is not constant with respect to the length of the timed state sequence. This happens because the sampling step is 0.2 and, thus, in each period the samples of the signal that belong to the set $\mathcal{O}(p1)$ are different.

An immediate application of the robustness estimate is the following. Assume that σ_1 is not generated as indicated in Fig. 2.2, but that it is the result of sampling

signal's time domain	number of samples	computation time (sec)	robustness
[0, 21.99]	110	0.00	0.097603
[0, 188.49]	943	0.03	0.097603
[0, 6283.2]	31,416	1.05	0.092065
[0, 219911.48]	1,099,558	37.61	0.091793

Table 3.2: Computation time for formula $\square(p_1 \rightarrow \langle \rangle_{(0.0, 1.0)}!p_1)$.

signal's time domain	number of samples	computation time (sec)	robustness
[0, 21.99]	110	0.16	0.238435
[0, 188.49]	943	2.35	0.237401
[0, 6283.2]	31,416	84.74	0.237149
[0, 219911.48]	1,099,558	out of memory	-

Table 3.3: Computation time for formula (3.12) for the first row and for formula $\square_{[0.0, T]}(\langle \rangle_{[0.0, 6.28]}(p_2 \wedge \langle \rangle_{[0.0, 3.14]}p_1))$ for the rest of the rows, where T is the maximum signal time minus 3π .

(monitoring) a physical quantity. In such a case, the sensors, which monitor the quantity, have a known experimentally determined accuracy. As an example, assume that the accuracy of the sensor in our case is ± 0.1 . Then, we immediately can infer that formulas (3.6), (3.11) and (3.12) are true over the monitored sampled signal since 0.1 (the sensor accuracy) is less than the robustness estimate of the formulas : 0.317274, 0.238435 and 0.238435 respectively. On the other hand, we cannot infer whether the signal σ_1 satisfies formulas (3.5) and (3.8) since 0.1 is greater than their robustness estimate of 0.097603 with respect to σ_1 . If we would like to logically infer something about the underlying continuous-time signal (not the sampled one), then one way to do so is to use the approach which is proposed in Chapter 5.

Now consider the following scenario. Signal s_1 is fed into a system which tries to track the input signal. The output of the system is signal s_2 in Fig. 3.4. For the sake of example, we set s_2 to be s_1 with a delay of 0.1 time units and a bounded noise of 0.1 magnitude. We monitor both signals with a constant sampling step of

0.2 time units. The result of the sampling process appears in Fig. 3.5. We would like to verify whether σ_2 is always within distance 0.25 of signal σ_1 and, moreover, if the difference of the two signals is greater than 0.25, then it should drop below 0.25 within 1 time unit. The above informal specification can be formally captured with the MTL formula $\Box(p_3 \vee (\neg p_3) \rightarrow \Diamond_{[0,1]} p_3)$ with $\mathcal{O}(p_3) = (-\infty, 0.25]$, which can be simplified to the formula $\Box(p_3 \vee \Diamond_{[0,1]} p_3)$. Now, if we test formula

$$\Box(p_3 \vee \Diamond_{[0,5]} p_3) \tag{3.13}$$

where the timing constraints are on the number of samples, over the signal $\sigma_3(i) = |\sigma_1(i) - \sigma_2(i)|$, we get

```
robustness : 0.038030
total running time : 0.020000 sec
```

If the timing constraints are stricter, that is,

$$\Box(p_3 \vee \Diamond_{[0,2]} p_3) \tag{3.14}$$

then the property does not hold any more (robustness -0.046525).

3.4 Related Research and Future Work

Since our research on robustness for temporal logic specifications spans many different research areas, the related literature is equally diverse. Here, we will just provide a few such references without attempting to be exhaustive.

Robustness in timed automata has been studied by several authors, for example [84, 98, 161, 10, 25, 178]. Out of the aforementioned literature, the work in [25] addresses the problem of robust temporal logic model checking of timed automata.

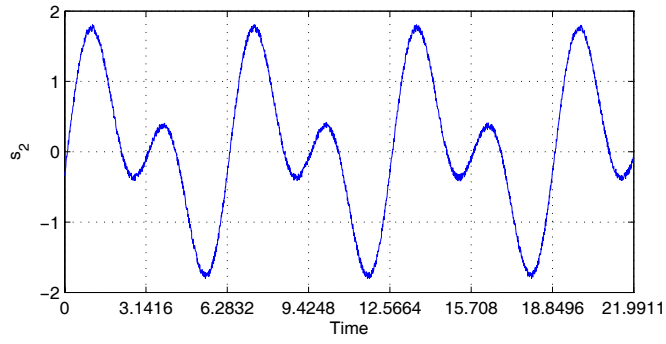


Figure 3.4: The signal s_2 .

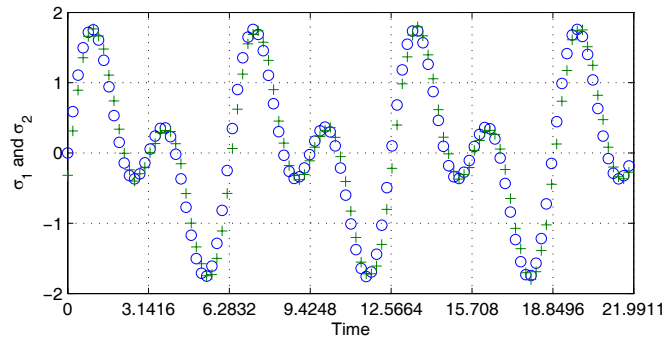


Figure 3.5: The samples of signal σ_1 are denoted by circles (o), while the samples of signal σ_2 by crosses (+).

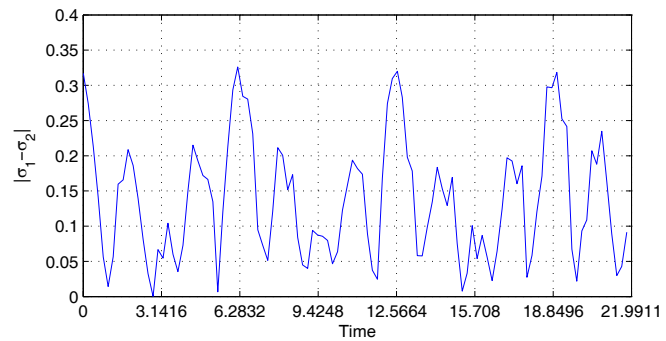


Figure 3.6: The discrete-time signal $\sigma_3(i) = |\sigma_1(i) - \sigma_2(i)|$.

The authors in [105] also consider robustness issues in MITL, but there the robustness is with respect to time. In hybrid systems, robustness issues have been analyzed in [67] and [98] among other works. We should point out that the authors in [84] and [98] define a notion of tube acceptance for timed and linear hybrid systems very similar to ours.

The authors in [134, 174, 119, 92] develop temporal logic monitoring algorithms for (Boolean valued) signals. In particular, in [134] the problem of MITL testing over continuous-time signals is addressed. The authors in [174] and [119] present algorithms for monitoring timed temporal logics over timed state sequences. Lastly, in [92] the authors develop efficient algorithms for LTL monitoring.

Our work on robustness has the same underlying motivation with quantitative temporal logics [50, 51, 96]. Namely, we need to determine the degree that a system (or signal) satisfies a specification in order to detect systems that are not robustly correct. However, our definitions for the robust semantics of the temporal logic operators are closer to the ones employed in multi-valued temporal logics [32, 33].

One open problem which is very interesting is whether we can get rid of the requirement in Section 3.2 that all the timed state sequences have the same timing function (or time-stamps). It might be possible to address this issue by introducing robustness also with respect to time. Another important extension to our framework is to allow Boolean signals along with signals that take values in non-trivial metric spaces. This will enable the possibility to express more complicated properties without sacrificing the very intuitive notion of robustness that we have introduced in Chapter 3.

Chapter 4

From Signals to Systems

Chapter 3 presented a new definition of robustness for propositional linear temporal logic specifications over signals. In this chapter, we introduce a similar notion for systems. Abstractly, a system is any collection of objects that interact with each other. This is a very general definition and it includes diverse systems such as computers, automobiles, ATMs, structures, electronic circuits and so on. Before we proceed to the definition of temporal logic robustness for systems, we need to introduce dynamical systems [29] and a notion of approximation between systems. The theory of approximation which we will be using in this thesis is based on the theory of approximate bisimulation relations [82] developed by Girard and Pappas.

4.1 Dynamical Systems

Historically, *dynamical systems* refer to physical systems such as mechanical, electrical and electromechanical systems whose behavior changes with time. The word “dynamical” is added to differentiate these systems from *static systems*, i.e., systems whose behavior might change spatially, but it is static with respect to time.

The mathematical formalisms which are employed in order to model dynamical systems are those of differential and difference equations. If a system is modeled using differential equations, then it is referred to as a *continuous-time dynamical system*, while if it is modeled using difference equations as a *discrete-time dynamical system*.

Definition 4.1.1 (Dynamical System). *A dynamical system is defined by a tuple $\Sigma = (\mathbb{T}, X, X^0, Y, U, P, f, g)$ where:*

- \mathbb{T} is the time domain,
- $X \subseteq \mathbb{R}^n$, for some $n \in \mathbb{N}$, is the state space of the system,
- $X^0 \subseteq X$ is the set of initial conditions,
- $Y \subseteq \mathbb{R}^m$, for some $m \in \mathbb{N}$, is the observation space,
- $U \subseteq \mathbb{R}^k$, for some $k \in \mathbb{N}$, is the input space,
- $P \subseteq \mathbb{R}^q$, for some $q \in \mathbb{N}$, is the parameter space,
- $f : \mathbb{T} \times X \times P \times U \rightarrow X$ is the map that governs the evolution of the system,
- $g \in \mathfrak{F}(X, Y)$ is the observation map.

The definition above includes systems that might have a discrete (N) or a continuous (R) time domain, uncertain or time varying parameters that take values in the set P , controllable inputs which can take values in U and an observation space Y . Note that both X and Y are metric spaces. The metric of choice for both spaces will be the Euclidean metric d_e .

The function f governs the behavior of the system. In this thesis, we will assume that given an initial condition $x^0 \in X^0$, an input signal $u : \mathbb{T} \rightarrow U$ and a time varying parameter $p : \mathbb{T} \rightarrow P$, the behavior of Σ is deterministic. That is, there

exists a unique continuous $x \in \mathfrak{F}(R, X)$ or discrete $x \in \mathfrak{F}(N, X)$ time signal that fully describes the status of the system with respect to time. In the following, we will refer to such signals as *state trajectories* of the system.

In detail, assume that Σ is a continuous-time dynamical system, i.e., $\mathbb{T} = R$, and that $x^0 \in X^0$, $u : R \rightarrow U$ and $p : R \rightarrow P$ are given. Note that the main difference between u and p is that we can control u whereas p is essentially a property of the system. Then, a state trajectory $x : R \rightarrow X$ of Σ is the unique solution¹ of the system of differential equations

$$\dot{x}(t) = f(t, x(t), p(t), u(t)) \quad (4.1)$$

such that $x(0) = x^0$ and a *trace* or *observable trajectory* is simply

$$y(t) = g(x(t))$$

Here, \dot{x} denotes the first order time derivative of the function x .

Similarly, if Σ is a discrete-time dynamical system, i.e., $\mathbb{T} = N$, and $x^0 \in X^0$, $u : N \rightarrow U$ and $p : N \rightarrow P$ are given. Then, a state trajectory $x : N \rightarrow X$ of Σ is the solution of the system of difference equations

$$x(i+1) = f(i, x(i), p(i), u(i)) \quad (4.2)$$

such that $x(0) = x^0$ and a *trace* or *observable trajectory* is simply

$$y(i) = g(x(i))$$

¹At this point, we simply assume that for the systems under consideration and for the given input and parameter signals, there exists a unique solution. Whenever required, we will impose conditions on the systems and on the input and parameter signals in order to guarantee that a unique solution exists.

Note that if the exact form of p is not known, but it is only known that p is a piecewise continuous function such that $p(t) \in P$ for $t \in \mathbb{T}$, then the system exhibits nondeterministic behavior. In other words, the solution of eq. (4.1) or (4.2) results into a family of trajectories instead of a unique trajectory.

Definition 4.1.1 presents a quite general class of systems. However in Part II, we will need to consider several subclasses of Σ with interesting structural properties. Of particular interest and with many practical applications is the class of linear systems [34]. In detail, if $t \in \mathbb{T}$, $f(x(t)) = Ax(t) + Bu(t)$, where $A \in \mathbb{R}^{n^2}$ and $B \in \mathbb{R}^{n \times k}$ are constant $n \times n$ and $n \times k$ matrices respectively, and $y(x(t)) = Cx(t)$, where $C \in \mathbb{R}^{m \times n}$ is a constant $m \times n$ matrix, then the system is called Linear Time Invariant (LTI) and we will characterize it by the tuple $\Sigma^{\text{LTI}} = (\mathbb{T}, X, X^0, Y, U, A, B, C)$. If f also depends on time, i.e., $f(t, x(t)) = A(t)x(t) + B(t)u(t)$, where now A and B are continuous matrix valued functions with respect to time, then the system is referred to as a Linear Time Varying (LTV) system. In this case, we use the symbol Σ^{LTV} to denote an LTV system and in order to indicate the dependence of the matrices on time t , we write $\Sigma^{\text{LTV}} = (\mathbb{T}, X, X^0, Y, U, A(t), B(t), C)$. On the other hand, if f depends on a piecewise continuous parameter function $p(t) \in P$, i.e., $f(x(t), p(t)) = A(p(t))x(t) + B(p(t))u(t)$, where A and B are again continuous matrix valued functions, then the system is referred to as a Linear Parameter Varying (LPV) system. We will characterize LPV systems by the tuple $\Sigma^{\text{LPV}} = (\mathbb{T}, X, X^0, Y, U, P, A(p), B(p), C)$. An LTI system that is derived from an LPV system for a specific constant parameter value $p_0 \in P$ will be denoted by $\Sigma^{\text{LPV}}(p_0) = (\mathbb{T}, X, X^0, Y, U, A(p_0), B(p_0), C)$.

A further constrained class of linear systems, is the class of autonomous or closed-loop linear systems. Informally, autonomous systems are systems whose dynamics do not depend on external inputs. For example, an autonomous LTI system has dynamics of the form $f(x(t)) = Ax(t)$. We will denote closed-loop systems using an

overline, e.g., $\overline{\Sigma}^{\text{LTI}} = (\mathbb{T}, X, X^0, Y, A, C)$ and similarly for $\overline{\Sigma}$, $\overline{\Sigma}^{\text{LPV}}$ and $\overline{\Sigma}^{\text{LTI}}$. Any system whose dynamics is not linear will be referred to as a *nonlinear system*. If, moreover, f is built upon indicator functions, then we will refer to it as a *hybrid system*.

Now, we introduce two notions of language for a dynamical system Σ in order to present some results on approximate bisimulation relations in the next section and to be able to talk about system verification in Part II. Our definitions follow closely the definitions of languages for signals and logical formulas. Given a dynamical system Σ , its internal language, which consists of all its state trajectories, is defined to be the set

$$\Lambda(\Sigma) = \{x \in \mathfrak{F}(\mathbb{T}, X) \mid x \text{ is a solution of (4.1) or (4.2)}\}$$

under an input signal $u \in \mathfrak{F}(\mathbb{T}, U)$ and a parameter signal $p \in \mathfrak{F}(\mathbb{T}, P)$

Again, we consider only combinations of systems-input signals-parameter signals such that a solution exists. Then, the language of Σ is simply the image of $\Lambda(\Sigma)$ through the map g , that is, $\mathcal{L}(\Sigma) = g(\Lambda(\Sigma))$.

In the case of discrete-time dynamical systems, we also need to consider the set of all timed state sequences that result by pairing the observation trajectories of Σ with a timing function. Formally, if $\tau \in \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$ is a timing function and $\Sigma = (N, X, X^0, Y, U, P, f, g)$ is a discrete-time dynamical system, then $TSS^\tau(\Sigma) = \{(y, \tau) \mid y \in \mathcal{L}(\Sigma)\}$. Something similar can be defined for continuous-time dynamical systems whose observation trajectories are being sampled. In detail, if $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ is a sampling function and $\Sigma = (R, X, X^0, Y, U, P, f, g)$ is a continuous-time dynamical system, then $TSS^\tau(\Sigma) = \{(y \circ \tau, \tau) \mid y \in \mathcal{L}(\Sigma)\}$. Finally, in order to be in accordance with the notation of the language of an MTL formula and, also,

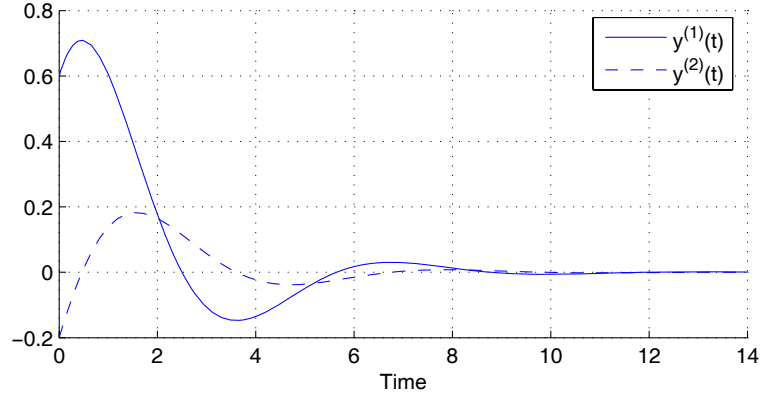


Figure 4.1: Example 4.1.1 : the observation trajectory $y(t)$ with respect to time.

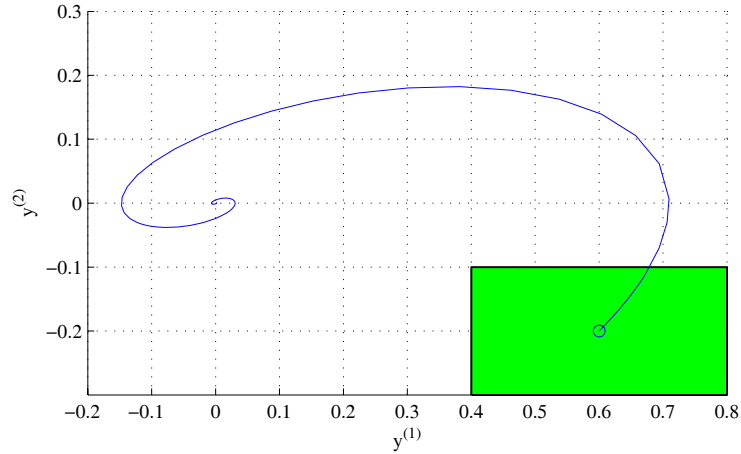


Figure 4.2: Example 4.1.1 : the observation trajectory $y(t)$ in phase space.

in order to treat uniformly discrete-time and sampled continuous-time dynamical systems, we introduce the notation $\mathcal{L}^\tau(\Sigma) = \{y \mid (y, \tau) \in TSS^\tau(\Sigma)\}$.

In the following we present some examples of dynamical systems.

Example 4.1.1. *Consider the autonomous nonlinear system*

$$\bar{\Sigma}_a = (R_a, \mathbb{R}^2, X_a^0, \mathbb{R}^2, f_a, g_a)$$

where $R_a = [0, 14] \subseteq \mathbb{R}_{\geq 0}$, $X_a^0 = [0.4, 0.8] \times [-0.3, -0.1]$, $g_a = \mathbf{id}$ (the identity

function) and

$$f_a(x(t)) = \begin{bmatrix} 0.05 \sin^2(x^{(2)}(t))x^{(1)}(t) - 2.5x^{(2)}(t) \\ 0.5x^{(1)}(t) - x^{(2)}(t) \end{bmatrix}$$

The observation trajectories $y(t) = x(t)$ of the system for the initial condition $x^0 = [0.6 \ -0.2]^T$ appear in Fig. 4.1, while the phase space appears in Fig. 4.2.

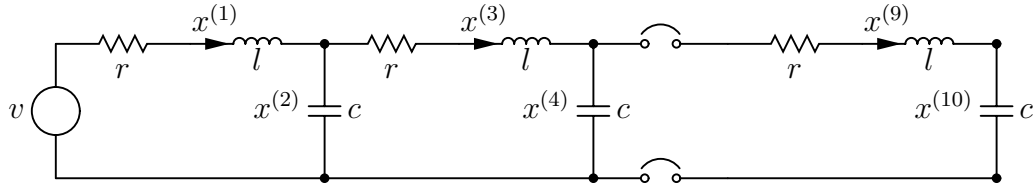


Figure 4.3: A ladder network representing a transmission line with 5 sections.

Example 4.1.2. *As an example of an LTI system*

$$\Sigma_b^{\text{LTI}} = (R_b, \mathbb{R}^{10}, X_b^0, \mathbb{R}^5, \mathbb{R}, A_b, b_b, C_b)$$

consider the RLC circuit in Fig. 4.1.2. Such circuits are used to represent high voltage transmission lines, where the requirement is the protection of the line against traveling waves, or the interconnect in ultra-deep submicron integrated circuits, where we have to study the interconnect delay. Under the assumption that the values of r , l and c are constant and known, we can easily derive (see for example [106]) a set of linear differential equations which form the state space representation of the RLC circuit. In detail, the system dynamics are given by the system

$$\dot{x}(t) = A_b x(t) + b_b u(t)$$

where A is defined in Fig. 4.1.2 and

$$A_b = \begin{bmatrix} -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{c} & 0 & -\frac{1}{c} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{c} & 0 & -\frac{1}{c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{c} & 0 & -\frac{1}{c} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{c} & 0 & -\frac{1}{c} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{c} & 0 \end{bmatrix}$$

Figure 4.4: The matrix A_b of Example 4.1.2.

$$C_b = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.5: The matrix C of Example 4.1.2.

$$b_b = \left[\frac{1}{l} \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \right]^T.$$

The state vector x consists of the currents in the inductances $(x^{(1)}, x^{(3)}, \dots, x^{(9)})$ and the voltages across the capacitances $(x^{(2)}, x^{(4)}, \dots, x^{(10)})$. The observation function is a linear map C (see Fig. 4.1.2) such that $y(t) = C_b x(t) = [x^{(2)}(t) \dots x^{(10)}(t)]^T$. Usually, the goal of the analysis of such systems is to study the transient behavior of the circuit for specific parameter values and initial conditions under a unit step input function, i.e., $u(t) = 1$ for all $t \in R$. A trajectory of the system under the parameters $r = 2.5$, $l = 1.25$, $c = 3.75 \cdot 10^{-3}$ and with initial conditions $x^0 = [0 \dots 0]^T$ and time domain $R_b = [0, 6]$ appears in Fig. 4.6.

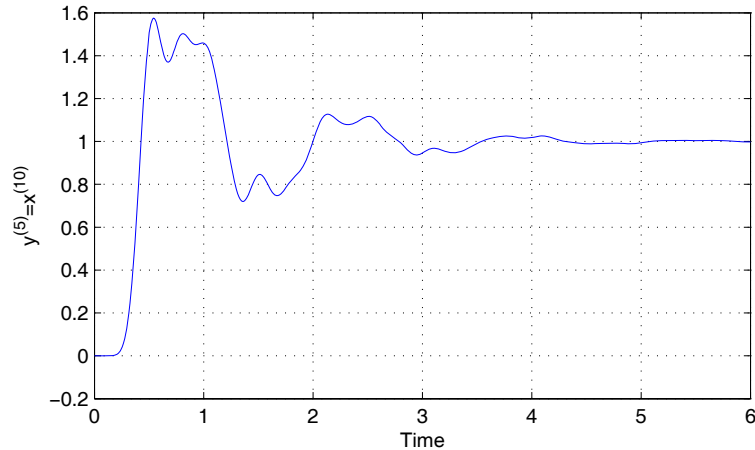


Figure 4.6: Example 4.1.2 : the observation trajectory $y^{(5)} = x^{(10)}$ with respect to time.

4.2 Approximate (Bi)Simulation Relations

Simulation and bisimulation relations [142] were developed as a way to reduce the size of state machines and automata mainly for verification purposes [41]. Intuitively, a simulation relation between two systems requires that if a system has an observable behavior, than the other system should also have the same observable behavior. One interesting connection between simulation relations and temporal logics is that simulation relations preserve linear temporal logic properties (with discrete-time semantics) [168].

Approximate simulation and bisimulation relations [82] for systems with metric spaces, as opposed to (exact) simulation and bisimulation relations [152], do not require that two observable trajectories are identical, but that they remain always close enough. In the following, we review some definitions and results from [82] and [81] which are going to be used in the subsequent chapters.

Definition 4.2.1 (Simulation Relation). *A relation $\mathcal{S}_\delta \subseteq X_1 \times X_2$ is a δ -approximate simulation relation of $\Sigma_1 = (\mathbb{T}, X_1, X_1^0, Y, U_1, f_1, g_1)$ by $\Sigma_2 = (\mathbb{T}, X_2, X_2^0, Y, U_2, f_2, g_2)$*

if for all $(x_1^0, x_2^0) \in \mathcal{S}_\delta$,

1. $d_e(y_1^0, y_2^0) = d_e(g_1(x_1^0), g_2(x_2^0)) \leq \delta$
2. For all $x_1 \in \Lambda(\Sigma_1)$ such that $x_1(0) = x_1^0$ there exists an $x_2 \in \Lambda(\Sigma_2)$ such that $x_2(0) = x_2^0$ and $\forall t \in \mathbb{T}$, $(x_1(t), x_2(t)) \in \mathcal{S}_\delta$.

A dynamical system Σ_1 is δ -approximately simulated by a dynamical system Σ_2 (noted $\Sigma_1 \preceq_\delta \Sigma_2$) if for all $x_1^0 \in X_1^0$, there exists $x_2^0 \in X_2^0$ such that $(x_1^0, x_2^0) \in \mathcal{S}_\delta$.

Informally, Definition 4.2.1 states that for each observable trajectory y_1 of Σ_1 there exists an observable trajectory y_2 of Σ_2 such that their distance at each point in time remains bounded by δ . Notice that the observation space for the two systems in the definition is the same, i.e., Y . Now, if a relation is a δ -approximate simulation relation of Σ_1 by Σ_2 and, also, a δ -approximate simulation relation of Σ_2 by Σ_1 , then it is called a δ -approximate bisimulation relation.

Definition 4.2.2 (Bisimulation Relation). *A relation $\mathcal{B}_\delta \subseteq X_1 \times X_2$ is an approximate bisimulation relation of precision δ between $\Sigma_1 = (\mathbb{T}, X_1, X_1^0, Y, U_1, f_1, g_1)$ and $\Sigma_2 = (\mathbb{T}, X_2, X_2^0, Y, U_2, f_2, g_2)$ if for all $(x_1^0, x_2^0) \in \mathcal{B}_\delta$,*

1. $d_e(y_1^0, y_2^0) = d_e(g_1(x_1^0), g_2(x_2^0)) \leq \delta$
2. For all $x_1 \in \Lambda(\Sigma_1)$ such that $x_1(0) = x_1^0$ there exists an $x_2 \in \Lambda(\Sigma_2)$ such that $x_2(0) = x_2^0$ and $\forall t \in \mathbb{T}$, $(x_1(t), x_2(t)) \in \mathcal{B}_\delta$.
3. For all $x_2 \in \Lambda(\Sigma_2)$ such that $x_2(0) = x_2^0$ there exists an $x_1 \in \Lambda(\Sigma_1)$ such that $x_1(0) = x_1^0$ and $\forall t \in \mathbb{T}$, $(x_1(t), x_2(t)) \in \mathcal{B}_\delta$.

Two systems Σ_1 and Σ_2 are bisimilar with precision δ (noted $\Sigma_1 \sim_\delta \Sigma_2$) if for all $x_1^0 \in X_1^0$, there exists $x_2^0 \in X_2^0$ such that $(x_1^0, x_2^0) \in \mathcal{B}_\delta$ and vice versa.

One way to think about approximate bisimulation relations is to observe that they provide us with a quantifiable bound on how far away are two systems from being bisimilar. Even though approximate bisimulation relations cannot be computed, they can be approximated using the notion of bisimulation functions. The relationship between approximate bisimulation relations and bisimulation functions is that the former can be characterized by the level sets of the latter.

Definition 4.2.3 (Bisimulation Function). *A function $\mathcal{F} : X_1 \times X_2 \rightarrow \overline{\mathbb{R}}_{\geq 0}$ is a bisimulation function between Σ_1 and Σ_2 if for all $\delta \geq 0$*

$$\mathcal{B}_\delta = \{(x_1, x_2) \in X_1 \times X_2 \mid \mathcal{F}(x_1, x_2) \leq \delta\}$$

is a δ -approximate bisimulation relation between Σ_1 and Σ_2 .

An interesting result proved in [82] is that we can compute a tight upper bound δ such that $\Sigma_1 \sim_\delta \Sigma_2$ by solving two games.

Theorem 4.2.1. *Let \mathcal{F} be a bisimulation function between Σ_1 and Σ_2 and*

$$\delta \geq \max \left\{ \sup_{x_1^0 \in X_1^0} \inf_{x_2^0 \in X_2^0} \mathcal{F}(x_1^0, x_2^0), \sup_{x_2^0 \in X_2^0} \inf_{x_1^0 \in X_1^0} \mathcal{F}(x_1^0, x_2^0) \right\}.$$

If δ has finite a value, then $\Sigma_1 \sim_\delta \Sigma_2$.

The following theorem, whose proof can be found in [81], provides a characterization of bisimulation functions.

Theorem 4.2.2. *Let $\Sigma_1 = (R, X_1, X_1^0, Y, U_1, f_1, g_1)$ and $\Sigma_2 = (R, X_2, X_2^0, Y, U_2, f_2, g_2)$ be two time invariant dynamical systems without uncertain parameters and with the following additional constraints for $i = 1, 2$:*

- X_i^0, U_i are compact sets,

- $u_i \in \mathfrak{F}(R, U_i)$ is a measurable function,
- f_i is Lipschitz continuous,
- for all $x_i \in X_i$, $f_i(x_i, U_i)$ is a convex set, and
- g_i is continuous.

Let $\mathcal{V} : X_1 \times X_2 \rightarrow \overline{\mathbb{R}}_{\geq 0}$ be a continuously differentiable function and $\alpha \geq 0$. If for all $(x_1, x_2) \in X_1 \times X_2$ we have

$$\mathcal{V}(x_1, x_2) \geq d_e^2(g_1(x_1), g_2(x_2)) \quad (4.3)$$

and for all $(x_1, x_2) \in X_1 \times X_2$ such that $\mathcal{V}(x_1, x_2) \geq \alpha^2$, we have

$$\sup_{u_1 \in U_1} \inf_{u_2 \in U_2} \nabla \mathcal{V}(x_1, x_2) \cdot \begin{bmatrix} f_1(x_1, u_1) \\ f_2(x_2, u_2) \end{bmatrix} \leq 0 \quad (4.4)$$

$$\sup_{u_2 \in U_2} \inf_{u_1 \in U_1} \nabla \mathcal{V}(x_1, x_2) \cdot \begin{bmatrix} f_1(x_1, u_1) \\ f_2(x_2, u_2) \end{bmatrix} \leq 0 \quad (4.5)$$

Then, $\mathcal{F}(x_1, x_2) = \max\{\sqrt{\mathcal{V}(x_1, x_2)}, \alpha\}$ is a bisimulation function between Σ_1 and Σ_2 .

Theorem 4.2.2 essentially states that given two time invariant dynamical systems without uncertain parameters, the distance between their observations is always bounded (eq. (4.3)) and, moreover, it cannot increase during the parallel evolution of the systems (eq. (4.4) and (4.5)). The structure of \mathcal{F} can be interpreted as follows. The function \mathcal{V} bounds the transient dynamics of the two systems, while α accounts for the error in the asymptotic behavior of the systems and thus it does not depend on the initial conditions of the systems.

Effective characterizations of bisimulation functions have been proposed for linear continuous-time dynamical systems based on a set of linear matrix inequalities [77, 81] and for nonlinear dynamical systems based on sum of squares programs [78]. Both characterizations can be interpreted in terms of convex optimization leading to efficient algorithms for the computation of bisimulation functions.

When we consider autonomous continuous-time dynamical systems, Theorem 4.2.2 reduces to the following corollary.

Corollary 4.2.1. *Let $\bar{\Sigma}_1 = (R, X_1, X_1^0, Y, f_1, g_1)$ and $\bar{\Sigma}_2 = (R, X_2, X_2^0, Y, f_2, g_2)$ be two autonomous time invariant dynamical systems without uncertain parameters and with the following additional constraints for $i = 1, 2$: (1) X_i^0 is a compact set, (2) f_i is Lipschitz continuous and (3) g_i is continuous. Let $\mathcal{F} : X_1 \times X_2 \rightarrow \bar{\mathbb{R}}_{\geq 0}$ be a continuously differentiable function such that for all $(x_1, x_2) \in X_1 \times X_2$ we have*

$$\mathcal{F}(x_1, x_2) \geq d_e(g_1(x_1), g_2(x_2)) \quad (4.6)$$

$$\frac{\partial \mathcal{F}}{\partial x_1}(x_1, x_2) \cdot f_1(x_1) + \frac{\partial \mathcal{F}}{\partial x_2}(x_1, x_2) \cdot f_2(x_2) \leq 0 \quad (4.7)$$

Then, \mathcal{F} is a bisimulation function between $\bar{\Sigma}_1$ and $\bar{\Sigma}_2$.

In the case of autonomous continuous-time dynamical systems with uncertain parameters, as the ones we consider in Chapter 6, a bisimulation function is any function that satisfies the conditions of the following theorem.

Theorem 4.2.3. *Let $\bar{\Sigma}_1 = (R, X_1, X_1^0, Y, P_1, f_1, g_1)$ and $\bar{\Sigma}_2 = (R, X_2, X_2^0, Y, P_2, f_2, g_2)$ be two autonomous dynamical systems with uncertain parameters and with the following additional constraints for $i = 1, 2$: (1) X_i^0 is a compact set, (2) f_i is Lipschitz continuous, and (3) g_i is continuous. Let $\mathcal{F} : X_1 \times X_2 \rightarrow \bar{\mathbb{R}}_{\geq 0}$ be a*

continuously differentiable function such that for all $(x_1, x_2) \in X_1 \times X_2$ we have

$$\mathcal{F}(x_1, x_2) \geq d_e(g_1(x_1), g_2(x_2)) \quad (4.8)$$

$$\forall p_1 \in P_1 \cdot \forall p_2 \in P_2 \cdot \frac{\partial \mathcal{F}}{\partial x_1}(x_1, x_2) \cdot f_1(x_1, p_1) + \frac{\partial \mathcal{F}}{\partial x_2}(x_1, x_2) \cdot f_2(x_2, p_2) \leq 0 \quad (4.9)$$

Then, \mathcal{F} is a bisimulation function between $\bar{\Sigma}_1$ and $\bar{\Sigma}_2$.

4.3 Robustness of MTL Specifications for Systems

4.3.1 Continuous Time

In system verification, we are usually interested in answering the question whether each observable trajectory of a continuous-time dynamical system Σ satisfies a temporal logic formula ϕ . In other words, whether $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\phi, \mathcal{O})$. Note that if the distance between the two sets $\mathcal{L}(\Sigma)$ and $\mathcal{L}(\neg\phi, \mathcal{O})$ is greater than zero, then the set inclusion $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\phi, \mathcal{O})$ holds.

Definition 4.3.1 (Distance between Sets). *Let S_1, S_2 be subsets of a set X which is equipped with a metric d . Then, the distance between S_1 and S_2 is defined as*

$$\mathbf{dist}_d(S_1, S_2) = \inf\{d(x_1, x_2) \mid x_1 \in S_1, x_2 \in S_2\}$$

Note that if the two sets intersect, then their distance is zero. With the previous definition at hand, it is straightforward to define the robustness degree of a system with respect to an MTL specification in a similar fashion to the robustness degree of a signal (Definition 3.1.1).

Definition 4.3.2 (Continuous-Time Robustness Degree). *Given a continuous-time dynamical system Σ , an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$ and an observation map*

$\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, the robustness degree $\varepsilon \in \overline{\mathbb{R}}_{\geq 0}$ of Σ with respect to the formula ϕ is $\varepsilon = \mathbf{dist}_\rho(\mathcal{L}(\Sigma), \mathcal{L}(\neg\phi, \mathcal{O}))$.

According to our definition of robustness, the greater the distance between the sets $\mathcal{L}(\Sigma)$ and $\mathcal{L}(\neg\phi, \mathcal{O})$, the greater the robustness of the system. Unlike our definition of the robustness degree for signals, the robustness degree for systems does not provide us with a measure of how robustly Σ does not satisfy ϕ when $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\neg\phi, \mathcal{O})$.

Now, assume that we are given two systems Σ_1 and Σ_2 such that Σ_2 simulates Σ_1 with precision δ . Then, for any trajectory in $\mathcal{L}(\Sigma_1)$ there exists a trajectory in $\mathcal{L}(\Sigma_2)$ such that their distance remains bounded by δ . Thus, it is easy to see that for any $\delta' \geq \delta$, we have $\mathcal{L}(\Sigma_1) \subseteq E_\rho(\mathcal{L}(\Sigma_2), \delta')$. Therefore, if $\varepsilon = \mathbf{dist}_\rho(\mathcal{L}(\Sigma_2), \mathcal{L}(\neg\phi, \mathcal{O})) > \delta'$, then we can conclude that Σ_1 satisfies ϕ since $E_\rho(\mathcal{L}(\Sigma_2), \delta') \subset E_\rho(\mathcal{L}(\Sigma_2), \varepsilon)$ and $E_\rho(\mathcal{L}(\Sigma_2), \varepsilon) \cap \mathcal{L}(\neg\phi, \mathcal{O}) = \emptyset$. Formally, we can state the following result.

Proposition 4.3.1. *For any formula $\phi \in MTL_{\mathbb{B}}(AP)$ and map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, if $\Sigma_1 \preceq_\delta \Sigma_2$ and $\mathbf{dist}_\rho(\mathcal{L}(\Sigma_2), \mathcal{L}(\neg\phi, \mathcal{O})) > \delta$, then $\mathcal{L}(\Sigma_1) \subseteq \mathcal{L}(\phi, \mathcal{O})$.*

The next question that we have to answer is how do we check the robustness of a system with respect to an MTL formula? It is obvious that the quantity $\mathbf{dist}_\rho(\mathcal{L}(\Sigma_2), \mathcal{L}(\neg\phi, \mathcal{O}))$ cannot be computed. Instead, we can check the inclusion $E_\rho(\mathcal{L}(\Sigma_2), \delta) \subseteq \mathcal{L}(\phi, \mathcal{O})$. This will be answered in Chapter 6. However, this is not the only way one can determine whether $\mathbf{dist}_\rho(\mathcal{L}(\Sigma_2), \mathcal{L}(\neg\phi, \mathcal{O})) > \delta$ holds. The following corollary is immediate from Theorem 3.1.2.

Corollary 4.3.1. *Consider a formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, a number $\delta > 0$ and set $\phi' = \mathbf{pos}(\mathbf{nnf}(\phi))$, then $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\phi', \mathcal{O}_\delta^\varepsilon)$ implies $E_\rho(\mathcal{L}(\Sigma), \delta) \subseteq \mathcal{L}(\phi', \mathcal{O})$ for any continuous-time dynamical system Σ .*

The advantage of Corollary 4.3.1 is that the problem $\mathcal{L}(\Sigma) \subseteq \mathcal{L}(\phi', \mathcal{O}_\delta^\varepsilon)$ could be

solved by any method that can verify a system Σ against an MTL specification ϕ . The next corollary reviews the results of this section.

Corollary 4.3.2. *For any formula $\phi \in MTL_{\mathbb{B}}(AP)$ and map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, if $\Sigma_1 \preceq_{\delta} \Sigma_2$ and*

1. $E_{\rho}(\mathcal{L}(\Sigma_2), \delta) \subseteq \mathcal{L}(\phi, \mathcal{O})$ or
2. $\mathcal{L}(\Sigma_2) \subseteq \mathcal{L}(\mathbf{pos}(\mathbf{nnf}(\phi)), \mathcal{O}_{\delta}^e)$,

then $\mathcal{L}(\Sigma_1) \subseteq \mathcal{L}(\phi, \mathcal{O})$.

4.3.2 Discrete Time

In this section, we briefly restate the results of Section 4.3.1 for discrete-time or sampled continuous-time dynamical systems.

Definition 4.3.3 (Discrete Time Robustness Degree). *Given a dynamical system Σ , a timing function $\tau \in \mathfrak{F}^{\uparrow}(N, \mathbb{R}_{\geq 0})$, an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$ and a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, the robustness degree $\varepsilon \in \overline{\mathbb{R}}_{\geq 0}$ of Σ with respect to the formula ϕ is $\varepsilon = \mathbf{dist}_{\hat{\rho}}(\mathcal{L}^{\tau}(\Sigma), \mathcal{L}^{\tau}(\neg\phi, \mathcal{O}))$.*

The following result holds for systems that have been sampled with the same sampling function or that have the same timing function.

Proposition 4.3.2. *For any formula $\phi \in MTL_{\mathbb{B}}(AP)$ and map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, if $\Sigma_1 \preceq_{\delta} \Sigma_2$ and*

1. $E_{\hat{\rho}}(\mathcal{L}^{\tau}(\Sigma_2), \delta) \subseteq \mathcal{L}^{\tau}(\phi, \mathcal{O})$ or
2. $\mathcal{L}^{\tau}(\Sigma_2) \subseteq \mathcal{L}^{\tau}(\mathbf{pos}(\mathbf{nnf}(\phi)), \mathcal{O}_{\delta}^e)$

with $\tau \in \mathfrak{F}^{\uparrow}(N, \mathbb{R}_{\geq 0})$, then $\mathcal{L}^{\tau}(\Sigma_1) \subseteq \mathcal{L}^{\tau}(\phi, \mathcal{O})$.

Part II

Applications

Chapter 5

Continuous-Time Satisfiability by Discrete-Time Reasoning

5.1 Introduction and Problem Formulation

The discrete-time robust semantics for MTL formulas have at least one important application. Given a continuous-time signal s and a timed state sequence $\mu = (\sigma, \hat{\tau})$ such that $\sigma = s \circ \hat{\tau}$, we can determine the relationship between $\llbracket \phi, \mathcal{O} \rrbracket_C(s)$ and $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu)$. This is an important problem since the timing (or better the *sampling*) function $\hat{\tau}$ may not just change the satisfiability of a formula ϕ with respect to a signal s , but also the validity of the formula [157]. In this section, we develop conditions for the signals in the set $\mathfrak{F}(R, X)$ and the sampling function $\hat{\tau}$ which can guarantee the equality $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu) = \llbracket \phi, \mathcal{O} \rrbracket_C(s)$ for MITL formulas. Formally, we address the following problem in Section 5.3.

Problem 5.1.1. *Given a formula $\phi \in MTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a continuous-time signal $s \in \mathfrak{F}(R, X)$, find a set of conditions and a sampling function $\hat{\tau} \in \mathfrak{F}^{\uparrow}(N, R)$ that will guarantee the equality $\llbracket \phi, \mathcal{O} \rrbracket_C(s) = \llbracket \phi, \mathcal{O} \rrbracket_D(\mu)$, where $\mu =$*

$(s \circ \hat{\tau}, \hat{\tau})$.

In Section 3.2, we developed a semantic approach for approximating the robustness degree of a timed state sequence with respect to an MTL formula. This approach enables the computation of a robustness estimate for finite timed state sequences. An important question that arises especially in testing and verification (see for example Chapter 6) is whether we can use the discrete-time robustness estimate in order to infer the value of the continuous-time robustness estimate of the underlying continuous-time signal. That is, we address the following problem.

Problem 5.1.2. *Given a formula $\phi \in MTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a continuous-time signal $s \in \mathfrak{F}(R, X)$, find a set of conditions and a sampling function $\hat{\tau} \in \mathfrak{F}^{\uparrow}(N, R)$ such that the value of $\llbracket \phi, \mathcal{O} \rrbracket_C(s)$ can be bounded by a function of the value of $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu)$, where $\mu = (s \circ \hat{\tau}, \hat{\tau})$.*

This problem is addressed in Section 5.4, but first we present a condition on the dynamics of a signal s which is required for the solution of both problems.

5.2 Bounds on the Signal Values

In order to reason about the behavior of continuous-time signals using discrete-time methods, we need to derive conservative bounds on the divergence of the value of a signal s between two consecutive samples (for example i and $i + 1$). We do that by requiring that the state distance between any two points in time is bounded by a positive nondecreasing function \mathcal{E} which depends only on the time difference between these two points.

Assumption 5.2.1. *The signals in the set $\mathfrak{F}(R, X)$ satisfy the condition*

$$\forall t, t' \in R . d(s(t), s(t')) \leq \mathcal{E}(|t - t'|), \quad (5.1)$$

where $\mathcal{E} : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is a positive nondecreasing function.

Notice that in (5.1) the bound on the distance between two values of the signal depends on the sampling function $\hat{\tau}$. In particular, one parameter of the sampling function that we might wish to control is the *maximum sampling step*:

$$\Delta\hat{\tau} = \sup_{i \in \mathbb{N}_{>0}} \{\hat{\tau}(i) - \hat{\tau}(i-1)\}. \quad (5.2)$$

If, moreover, the sampling function $\hat{\tau}$ has a constant sampling rate α , then $\Delta\hat{\tau} = \alpha$. Thus, in this case the control parameter becomes the sampling rate α . In the next two sections, we develop conditions for $\Delta\hat{\tau}$ for two different fragments of MTL.

5.3 Sampling for MITL Satisfiability

The sampling function $\hat{\tau}$, i.e., the maximum sampling step $\Delta\tau$, must be such that the relationship between valid formulas in continuous and sampled semantics is maintained [157]. For example, it is easy to see that the formula $\Box_{[1,2]}p$ is true for any signal s if there is no sample in the interval $[1, 2]$. In order to avoid such situations, we must impose certain constraints to $\Delta\hat{\tau}$. But first, a slight modification of the timing constraints of the temporal operators is required.

In order to modify the timing constraints of the temporal operators in a consistent way, we must convert the input formula $\phi \in MTL_{\mathbb{B}}(AP)$ into Negation Normal Form (NNF). In the following we assume that the input formula is given directly in NNF. Similarly to [105], we strengthen MTL formulas by changing the timing requirements of a given formula ϕ . In detail, we introduce a function $\mathbf{str}_{\Delta\hat{\tau}} : MTL_{\mathbb{B}}^+(AP) \rightarrow MTL_{\mathbb{B}}^+(AP)$ that recursively operates on a formula ϕ and modifies the temporal

operators as follows

$$\begin{aligned}\mathbf{str}_{\Delta\hat{\tau}}(\phi_1 \mathcal{U}_{\mathcal{I}}\phi_2) &= \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{U}_{C_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \\ \mathbf{str}_{\Delta\hat{\tau}}(\phi_1 \mathcal{R}_{\mathcal{I}}\phi_2) &= \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{R}_{E_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2)\end{aligned}$$

while keeping the atomic propositions and constants the same, i.e., $\mathbf{str}_{\Delta\hat{\tau}}(\top) = \top$, $\mathbf{str}_{\Delta\hat{\tau}}(\perp) = \perp$, $\mathbf{str}_{\Delta\hat{\tau}}(p) = p$, $\mathbf{str}_{\Delta\hat{\tau}}(\neg p) = \neg p$, and simply recursing in the case of Boolean connectives, i.e., $\mathbf{str}_{\Delta\hat{\tau}}(\phi_1 \vee \phi_2) = \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \vee \mathbf{str}_{\Delta\hat{\tau}}(\phi_2)$ and $\mathbf{str}_{\Delta\hat{\tau}}(\phi_1 \wedge \phi_2) = \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \wedge \mathbf{str}_{\Delta\hat{\tau}}(\phi_2)$. In the above formulas, the metric d_1 is defined as follows: for any $t_1, t_2 \in \mathbb{R}$, $d_1(t_1, t_2) = |t_1 - t_2|$. The intuition behind the function $\mathbf{str}_{\Delta\hat{\tau}}$ is that a robust specification with respect to the atomic propositions must also be robust with respect to the timing constraints. For example, in order to determine the Boolean truth value of ϕ_2 in $\phi_1 \mathcal{R}_{\mathcal{I}}\phi_2$ for the whole interval \mathcal{I} in continuous-time, we must also consider the first samples after and before the interval $\hat{\tau}(i) + \mathcal{I}$.

Proposition 5.3.1. *For any $\phi \in MTL_{\mathbb{B}}^+(AP)$, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and $s \in \mathfrak{F}(R, X)$, $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ such that $\mu = (s \circ \hat{\tau}, \hat{\tau})$, we have that $\langle\langle \mathbf{str}_{\Delta\hat{\tau}}(\phi), \mathcal{O} \rangle\rangle_D(\mu, i) = \top$ implies $\langle\langle \phi, \mathcal{O} \rangle\rangle_D(\mu, i) = \top$.*

The next two assumptions guarantee the existence of at least one sampling point within each timing interval of the temporal operators.

Assumption 5.3.1. *Given a formula $\phi \in MTL_{\mathbb{B}}^+(AP)$, the sampling functions in the set $\mathfrak{F}^\dagger(N, R)$ satisfy the constraint:*

$$\Delta\hat{\tau} < \min_{\mathcal{I} \in (\mathfrak{J}(\mathbf{str}_{\Delta\hat{\tau}}(\phi)) \cup \mathfrak{J}(\phi))} \{\sup \mathcal{I} - \inf \mathcal{I}\}. \quad (5.3)$$

When R is bounded, the sampling functions in the set $\mathfrak{F}^\dagger(N, R)$ must also satisfy the constraint : $\sup R - \hat{\tau}(\max N) < \Delta\hat{\tau}$.

In the assumption above, $\mathfrak{I}(\phi)$ denotes the set of all timing constraints \mathcal{I} that appear in the temporal operators of an MTL formula ϕ . Notice that if there exists a singleton interval in the set $\mathfrak{I}(\phi)$, then the above assumption cannot be satisfied. This observation mandates the choice of the Metric Interval Temporal Logic (MITL) [5] as a specification language instead of MTL. MITL is a decidable fragment of MTL where the timing constraints \mathcal{I} of the temporal operators are not allowed to be singleton sets, i.e., it must be $\inf \mathcal{I} < \sup \mathcal{I}$. It is easy to see that with respect to the initial formula ϕ , Assumption 5.3.1 can be satisfied by the following constraint:

$$\Delta \hat{\tau} < 1/3 \min_{\mathcal{I} \in \mathfrak{I}(\phi)} \{\sup \mathcal{I} - \inf \mathcal{I}\}. \quad (5.4)$$

Lemma 5.3.1. *Consider a formula $\phi \in \text{MITL}_{\mathbb{B}}^+(AP)$ and a sampling function $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ and let Assumption 5.3.1 hold. Let $\mathcal{I} \in \mathfrak{I}(\phi)$. If for some $i \in N$, $\hat{\tau}(i) + \mathcal{I} \subseteq R$, then $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}) \neq \emptyset$.*

Whenever R is a bounded time interval, we have to impose additional constraints on the signal and the MITL formulas. That is, we require that all the intervals in $\mathfrak{I}(\phi)$ are bounded as it was initially suggested in [134]. This enables us to compute a minimum time $\mathbf{dur}(\phi)$ that guarantees in combination with Assumption 5.3.1 that there are no subformulas whose truth value was determined by the lack of sampling points. The computation of the minimum time $\mathbf{dur}(\phi)$ is performed recursively:

$$\begin{aligned} \mathbf{dur}(\alpha) &:= 0 && \text{for } \alpha \in \{p, \neg p, \top, \perp\} \\ \mathbf{dur}(\phi_1 \sim \phi_2) &:= \mathbf{dur}(\phi_1) \sqcup \mathbf{dur}(\phi_2) && \text{for } \sim \in \{\wedge, \vee\} \\ \mathbf{dur}(\phi_1 \mathcal{W}_{\mathcal{I}} \phi_2) &:= \sup \mathcal{I} + \mathbf{dur}(\phi_1) \sqcup \mathbf{dur}(\phi_2) && \text{for } \mathcal{W} \in \{\mathcal{U}, \mathcal{R}\} \end{aligned}$$

In particular, we would like to avoid the case where R is a bounded domain and

$(t + \mathcal{I}) \cap R \neq \emptyset$, but $t + \mathcal{I} \not\subseteq R$ and there is no sample in $t + {}_R\mathcal{I}$.

Example 5.3.1. Consider the sampling function $\hat{\tau}(i) = 0.5i$, i.e., $\Delta\hat{\tau} = 0.5$, and the formula $\phi = \square_{[2.2,4.2]}p$. Let the domain of the signal s be $R = [0, 2.4]$, then $N = \{0, 1, 2, 3, 4\}$ and $\hat{\tau}(N) = \{0, 0.5, 1, 1.5, 2\}$. Note that the constraints of Assumption 5.3.1 are satisfied, that is, $\Delta\hat{\tau} < 1/3(4.2 - 2.2)$ and $\sup R - \hat{\tau}(\max N) = 2.4 - 2 = 0.4 < \Delta\hat{\tau}$. The formula $\square_{[2.2,4.2]}p$ evaluates to \top simply because $\hat{\tau}^{-1}(0 + [2.2, 4.2]) = \hat{\tau}^{-1}([2.2, 4.2]) = \emptyset$. However, over the time interval $[2.2, 2.4]$ it might not be true that s satisfies ϕ .

In order to avert such situations, we must impose one additional constraint (when R is bounded). Namely, for a given formula ϕ and signal s , we let $\mathbf{dur}(\phi) < \sup R < +\infty$. In other words, both the domain of the signal and all the timing constraints in the formula are bounded from above and below. Now, assume that the temporal nesting depth of a formula ϕ is m and that a temporal subformula $\psi = \psi_1 \mathcal{W}_{\mathcal{I}_k} \psi_2$ of ϕ is at nesting depth k , where $\mathcal{W} \in \{\mathcal{U}, \mathcal{R}\}$. Let $\{\mathcal{I}_j\}_{m \geq j > k}$ be any sequence of timing constraints of nested temporal operators at higher nesting depths j than k . Informally, the temporal nesting depth of a formula ϕ is defined to be the maximum number of nested temporal operators and it is computed in a similar way to \mathbf{dur} where $\sup \mathcal{I}$ is replaced by 1. Then, for all $t \in [0, \sum_{j=k+1}^m \sup \mathcal{I}_j]$, we have $t + \mathcal{I}_k \subseteq R$ since $\sum_{j=k}^m \sup \mathcal{I}_j \leq \mathbf{dur}(\phi) < \sup R$. Therefore, $t + \mathcal{I}_k = t + {}_R\mathcal{I}_k$.

Example 5.3.2. Consider the formula $\phi = (p_1 \mathcal{U}_{[1,2]}p_2 \vee p_3 \mathcal{U}_{[3,4]}p_4) \mathcal{U}_{[4,6]}(p_5 \mathcal{U}_{[0,1]}p_6)$. Then, $\mathbf{dur}(\phi) = 10$ and the temporal nesting depth of ϕ is 2. All the possible sequences of timing constraints of nested temporal operators are : $\{[4, 6], [1, 2]\}$, $\{[4, 6], [3, 4]\}$ and $\{[4, 6], [0, 1]\}$. Let $\sup R > 10$ and consider the sequence $\{\mathcal{I}_2, \mathcal{I}_1\}$ where $\mathcal{I}_2 = [4, 6]$ and $\mathcal{I}_1 = [1, 2]$, then at nesting depth $k = 1$, for all $t \in [0, \sum_{j=2}^2 \sup \mathcal{I}_j] = [0, 6]$, we have $t + \mathcal{I}_1 = [t + 1, t + 2] \subseteq R$.

Assumption 5.3.2. *If the time domain R of the set of signals $\mathfrak{F}(R, X)$ is bounded, i.e., $\sup R < +\infty$, then for the formula $\phi \in \text{MITL}_{\mathbb{B}}^+(AP)$ under consideration the following conditions must hold : for all $\mathcal{I} \in \mathfrak{I}(\phi)$, we have $\sup \mathcal{I} < +\infty$ and, also, $\sup R > \text{dur}(\text{str}_{\Delta\hat{\tau}}(\phi))$.*

Lemma 5.3.2. *Consider a formula $\phi \in \text{MITL}_{\mathbb{B}}^+(AP)$ and a sampling function $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ and let Assumptions 5.3.1 and 5.3.2 hold. Let $\psi = \psi_1 \mathcal{W}_{\mathcal{I}_k} \psi_2$, where $\mathcal{W} \in \{\mathcal{U}, \mathcal{R}\}$, be a subformula of $\text{str}_{\Delta\hat{\tau}}(\phi)$ at nesting depth k and let $\{\mathcal{I}_j\}_{k>j}$ be any sequence of timing constraints of nested temporal operators at higher nesting depths $j > k$. If $I = \hat{\tau}^{-1}(T) \neq \emptyset$, where $T = [0, \sum_{j>k} \sup \mathcal{I}_j]$, then for all $i \in I$, we have $(\hat{\tau}(i) + {}_R\mathcal{I}_k) = (\hat{\tau}(i) + \mathcal{I}_k)$ and $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$.*

The above assumptions enable us to prove the following theorem.

Theorem 5.3.1. *Consider $\phi \in \text{MITL}_{\mathbb{B}}^+(AP)$, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, $s \in \mathfrak{F}(R, X)$, $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ and let Assumptions 5.2.1 to 5.3.2 hold. Then, $\llbracket \text{str}_{\Delta\hat{\tau}}(\phi), \mathcal{O} \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$ with $\mu = (s \circ \hat{\tau}, \hat{\tau})$ implies*

$$\forall t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R. \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) = \top \quad (5.5)$$

for any $i \in N$ which satisfies the conditions of Lemma 5.3.2.

We should remark that the conclusion (5.5) of Theorem 5.3.1 does not imply that the continuous-time Boolean signal $\mathcal{O}^{-1} \circ s$ satisfies the finite variability property as it is defined in [99]. It only states that there exists some interval in R of length at least $2\Delta\hat{\tau}$ such that the Boolean truth value of some atomic propositions remains constant. The following corollary is immediate from Theorem 5.3.1 and Propositions 3.2.2 and 5.3.1.

Corollary 5.3.1. *Consider $\phi \in MITL_{\mathbb{B}}^+(AP)$, $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, $s \in \mathfrak{F}(R, X)$, $\hat{\tau} \in \mathfrak{F}^\dagger(N, R)$ and let Assumptions 5.2.1–5.3.2 hold. Then, $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi), \mathcal{O} \rrbracket_D(\mu) > \mathcal{E}(\Delta\hat{\tau})$ with $\mu = (s \circ \hat{\tau}, \hat{\tau})$ implies $\llbracket \phi, \mathcal{O} \rrbracket_C(s) = \llbracket \phi, \mathcal{O} \rrbracket_D(\mu) = \top$.*

If the condition $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi), \mathcal{O} \rrbracket_D(\mu) > \mathcal{E}(\Delta\hat{\tau})$ fails, then in general we cannot infer anything about the relationship of the two semantics. Two strategies in order to guarantee the above condition would be (i) to reduce the size of the sampling step $\Delta\hat{\tau}$ or (ii) to devise an on-line monitoring procedure that can adjust real-time the sampling step according to the robustness estimate of a signal with respect to an MITL formula ϕ .

5.4 Sampling for MTL Robustness

Corollary 5.3.1 provides sufficient conditions for MITL formulas for semantic equality between the two different time domains, i.e., $\llbracket \phi, \mathcal{O} \rrbracket_C(s) = \llbracket \phi, \mathcal{O} \rrbracket_D(\mu)$. Another interesting question is whether we can relate the continuous and discrete-time robustness estimates. This is possible, but more stringent conditions on the MTL formula and the sampling function are required. Namely, we have to impose a constant sampling rate on the sampling function and, moreover, the timing constraints on the temporal operators must be closed intervals and must have sampling instants as bounds. Examples of such signals and MTL specifications can be found in Section 3.3.2. Note that in this case, we can allow punctual timing requirements, that is, \mathcal{I} can be a singleton set.

Assumption 5.4.1. *Given a formula $\phi \in MTL_{\mathbb{B}}(AP)$, we require that all $\mathcal{I} \in \mathfrak{I}(\phi)$ are of the form $\mathcal{I} = [\iota_1, \iota_2]$ where $\iota_1, \iota_2 \in \mathbb{Q}$ with $\iota_1 \leq \iota_2$ or $\mathcal{I} = [\iota, +\infty)$ where $\iota \in \mathbb{Q}$.*

Here, \mathbb{Q} denotes the set of rational numbers. Since the timing constraints \mathcal{I} have rational numbers as bounds, we can always find a common divisor α (or their greatest

common divisor) and use it as a sampling constant. This sampling constant guarantees that the corresponding sampling function $\hat{\tau}$ will always sample time instants that are at least on the boundaries of the required timing intervals.

Assumption 5.4.2. *Given a formula $\phi \in MTL_{\mathbb{B}}(AP)$ that satisfies Assumption 5.4.1, we construct a sampling function $\hat{\tau} \in \mathfrak{F}_c^\uparrow(N, R)$ with constant sampling rate α , where α is a common divisor of all the finite bounds of the temporal operators in $\mathfrak{I}(\phi)$.*

The following result is immediate if we rewrite the bounds of the time intervals in $\mathfrak{I}(\phi)$ as multiples of the constant α (for example, $\mathcal{I} = [\alpha i_1, \alpha i_2]$ for some $i_1 \leq i_2 \in \mathbb{N}$) and define the sampling function $\hat{\tau}$ to be $\hat{\tau}(i) = \alpha i$ for $i \in N$.

Lemma 5.4.1. *Consider a formula $\phi \in MTL_{\mathbb{B}}(AP)$ which satisfies assumption 5.4.1 and a sampling function $\hat{\tau} \in \mathfrak{F}_c^\uparrow(N, R)$ which satisfies Assumption 5.4.2. If for some $i \in N$, we have $(\hat{\tau}(i) + \mathcal{I}) \cap R \neq \emptyset$, then $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}) \neq \emptyset$.*

An implication of Lemma 5.4.1 is that if the signal s is of unbounded duration, i.e., $\sup R = +\infty$, then no other assumptions are required since we will always have enough sampling points in order to infer a robustness estimate for the formula. On the other hand, if the time domain of s is bounded, then we must impose additional constraints on the MTL formula ϕ or the time domain R as in Section 5.3.

Assumption 5.4.3. *Let $\hat{\tau} \in \mathfrak{F}_c^\uparrow(N, R)$. If the time domain R of the set of signals $\mathfrak{F}(R, X)$ is bounded, i.e., $\sup R < +\infty$, then for the formula $\phi \in MTL_{\mathbb{B}}(AP)$ under consideration at least one of the following two conditions must hold :*

1. *For all $\mathcal{I} \in \mathfrak{I}(\phi)$, we have $\sup \mathcal{I} < +\infty$ and, also, $\sup R > \mathbf{dur}(\phi) + \Delta \hat{\tau}$.*
2. *For all $\mathcal{I} \in \mathfrak{I}(\phi)$, we have $\min \mathcal{I} = 0$.*

Note that in the above assumption the second condition does not impose any requirements on the minimum duration of the continuous-time signal. Intuitively, the condition $0 \in \mathcal{I}$ guarantees that the set $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I})$ for $i \in N$ always contains at least one sampling point, namely, i . Similarly to Section 5.3, we can prove the following result.

Lemma 5.4.2. *Consider a formula $\phi \in MTL_{\mathbb{B}}(AP)$ and a sampling function $\hat{\tau} \in \mathfrak{F}_c^{\uparrow}(N, R)$ which satisfy Assumptions 5.4.1 to 5.4.3. Let $\psi = \psi_1 \mathcal{W}_{\mathcal{I}_k} \psi_2$, where $\mathcal{W} \in \{\mathcal{U}, \mathcal{R}\}$, be a subformula of ϕ at nesting depth k and let $\{\mathcal{I}_j\}_{k > j}$ be any sequence of timing constraints of nested temporal operators at higher nesting depths $j > k$. If $I = \hat{\tau}^{-1}(T) \neq \emptyset$, where $T = [0, \sum_{j > k} \sup \mathcal{I}_j]$, then for all $i \in I$, we have $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$ and, moreover, $\forall t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R$ we have $t +_R \mathcal{I}_k \neq \emptyset$.*

Before we proceed to state the main result of this section, we need to define one more translation function that operates on MTL formulas. In detail, we define a new translation function $\mathbf{mtc} : MTL_{\mathbb{B}}(AP) \rightarrow MTL_{\mathbb{B}}(AP)$ that recursively operates on a formula ϕ and modifies the temporal operators as follows

$$\begin{aligned} \mathbf{mtc}(\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2) &= \mathbf{mtc}(\phi_1) \overline{\mathcal{U}}_{\mathcal{I}} \mathbf{mtc}(\phi_2) \\ \mathbf{mtc}(\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2) &= \mathbf{mtc}(\phi_1) \overline{\mathcal{R}}_{\mathcal{I}} \mathbf{mtc}(\phi_2) \end{aligned}$$

Similarly to the function $\mathbf{str}_{\Delta\hat{\tau}}$, the Boolean connectives just recursively call \mathbf{mtc} and the atomic propositions and the constants remain the same, e.g., $\mathbf{mtc}(\top) = \top$, $\mathbf{mtc}(p) = p$, $\mathbf{mtc}(\neg\phi) = \neg\mathbf{mtc}(\phi)$ and $\mathbf{mtc}(\phi_1 \vee \phi_2) = \mathbf{mtc}(\phi_1) \vee \mathbf{mtc}(\phi_2)$. Here, \mathbf{mtc} stands for *matching* as it is defined for the until operator in [72] and, thus, we will refer to $\overline{\mathcal{U}}$ and $\overline{\mathcal{R}}$ as matching until and matching release respectively. The temporal operators $\overline{\mathcal{U}}$ and $\overline{\mathcal{R}}$ are derived operators defined as $\phi_1 \overline{\mathcal{U}}_{\mathcal{I}} \phi_2 = \phi_1 \mathcal{U}_{\mathcal{I}}(\phi_1 \wedge \phi_2)$ and $\phi_1 \overline{\mathcal{R}}_{\mathcal{I}} \phi_2 = \phi_1 \mathcal{R}_{\mathcal{I}}(\phi_1 \vee \phi_2)$. Intuitively, the formula $\phi_1 \overline{\mathcal{U}}_{\mathcal{I}} \phi_2$ requires that there exists

some time in \mathcal{I} such that both ϕ_2 and ϕ_1 are true and that for all previous time ϕ_1 holds. Note that the semantics of the matching versions of the temporal operators \diamond and \square are the same as their non-matching versions. The necessity for matching versions of the temporal operators will become apparent in the proof of Theorem 5.4.1. Now, using Assumptions 5.4.1 to 5.4.3, we can prove the following theorem.

Theorem 5.4.1. *Consider a formula $\phi \in MTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a continuous-time signal $s \in \mathfrak{F}(R, X)$ and a sampling function $\hat{\tau} \in \mathfrak{F}_c^\uparrow(N, R)$. Let Assumptions 5.2.1 and 5.4.1 to 5.4.3 hold and set $\mu = (s \circ \hat{\tau}, \hat{\tau})$. Then,*

$$\begin{aligned} \forall t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R. \\ \llbracket \mathbf{mtc}(\phi), \mathcal{O} \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq \llbracket \mathbf{mtc}(\phi), \mathcal{O} \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) \end{aligned} \quad (5.6)$$

for any $i \in N$ which satisfies the conditions of Lemma 5.4.2.

Theorem 5.4.1 allows us to bound the robustness estimate of a continuous-time signal with respect to an MTL formula ϕ . Note that the shortest the sampling constant $\Delta\hat{\tau} = \alpha$ of the timed state sequence is, the tighter are the bounds on the robustness estimate. However, since we cannot always assume that $\lim_{\Delta\hat{\tau} \rightarrow 0} \mathcal{E}(\Delta\hat{\tau}) = 0$ (see Example 5.5.2), we cannot make any further claims. Moreover, not only we can guarantee that the continuous-time signal s satisfies the specification ϕ when $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$, but also that the signal does not satisfy ϕ when $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu, i) < -\mathcal{E}(\Delta\hat{\tau})$.

Note that LTL is a fragment of MTL which satisfies the second condition of Assumption 5.4.3. Hence, the following corollary of Theorem 5.4.1 is particularly useful in the case of LTL formulas.

Corollary 5.4.1. *Consider a formula $\phi \in LTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$, a continuous-time signal $s \in \mathfrak{F}(R, X)$ and a sampling function $\hat{\tau} \in \mathfrak{F}_c^\uparrow(N, R)$. Let*

Assumption 5.2.1 hold and set $\sigma = s \circ \hat{\tau}$. Then,

$$\forall t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R.$$

$$\llbracket \mathbf{mtc}(\phi), \mathcal{O} \rrbracket_D(\sigma, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi, \mathcal{O} \rrbracket_C(s, t) \leq \llbracket \mathbf{mtc}(\phi), \mathcal{O} \rrbracket_D(\sigma, i) + \mathcal{E}(\Delta\hat{\tau}).$$

Note that LTL formulas, as opposed to MTL formulas, do not provide us with any information on how to sample the continuous-time signal. In this case, the shorter the sampling rate is, the better the approximation.

5.5 Examples

In this section, we demonstrate the proposed methodology with some examples. As mentioned in the introduction, we want to study the transient behavior of dynamical systems, thus all our examples study signals of bounded duration. The discrete-time signals under consideration could be the result of sampling a physical signal or a simulated one. The latter is meaningful in cases where we would like to use fewer sampled points for temporal logic testing, while simulating the actual trajectory with finer integration step. Since we analyze discrete-time signals of bounded duration, we can compute their robustness estimate with respect to an MTL formula ϕ using Algorithm 1.

First, we demonstrate that for certain classes of signals it is straightforward to construct a bounding function \mathcal{E} that satisfies the conditions of Assumption 5.2.1. For example, the function \mathcal{E} can be easily derived when a signal is Lipschitz continuous.

Definition 5.5.1 (Lipschitz Continuity). *Let (X, d) and (X', d') be two metric spaces. A function $f : X' \rightarrow X$ is called Lipschitz continuous if there exists a constant $L_f \geq 0$*

such that:

$$\forall x'_1, x'_2 \in X'. d(f(x'_1), f(x'_2)) \leq L_f d'(x'_1, x'_2). \quad (5.7)$$

The smallest constant L_f is called Lipschitz constant of the function f .

What we are actually interested in is Lipschitz continuity of a signal s with respect to time:

$$\forall t, t' \in R. d(s(t), s(t')) \leq L_s |t - t'|. \quad (5.8)$$

Any signal with bounded time derivative satisfies the above condition. Whenever only a number of values of the signal are available to us, instead of an analytical description, we can use methods from optimization theory in order to estimate a Lipschitz constant for the signal [177]. Moreover, if the signal s is the solution of an ordinary differential equation $\dot{s}(t) = f(s(t))$, where f is Lipschitz continuous with constant L_f , then it is always possible to estimate a constant L_s for eq. (5.8) when the time domain R of s is compact [130]. This estimate is very conservative and it cannot be employed in practical applications. However, it can be used as a local estimate for the Lipschitz constant at a sampling point i , i.e., for the time period $\hat{\tau}(i+1) - \hat{\tau}(i)$, in connection with an on-line monitoring algorithm.

In all the examples that follow, we set $X = \mathbb{R}$ and $d(x_1, x_2) = |x_1 - x_2|$. The first example exploits the fact that the derivative of the signal can be bounded.

Example 5.5.1. Assume that we are given a discrete-time representation σ_1 (Fig. 2.2) of the continuous-time signal s_1 (Fig. 2.1) which has constant sampling step of magnitude 0.2, i.e., $\Delta\hat{\tau}_1 = 0.2$. We are also provided with the constraint $\mathcal{E}_1(t) = 3t$ (notice that $|\dot{s}_1(t)| \leq |\cos t| + 2|\cos 2t| \leq 1 + 2 = 3$ for all $t \in R$, therefore s_1 is Lipschitz continuous with $L_{s_1} = 3$). We would like to test whether the underlying continuous-time signal s_1 satisfies the specification $\phi_1 = \square_{[0, 9\pi/2]}(p_{11} \rightarrow \diamond_{[\pi, 2\pi]} p_{12})$, with $\mathcal{O}(p_{11}) = \mathbb{R}_{\geq 1.5}$ and $\mathcal{O}(p_{12}) = \mathbb{R}_{\leq -1}$. Notice that the sampling function $\hat{\tau}_1$ sat-

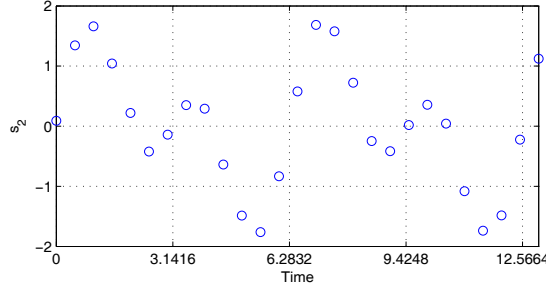


Figure 5.1: The sampled signal σ_2 generated by sampling the continuous-time signal $s_2(t) = \sin(t) + \sin(2t) + w(t)$, where $|w(t)| \leq 0.1$, with constant sampling period 0.5. In this case, it is $|s_2(t_1) - s_2(t_2)| \leq L_{s_1}|t_1 - t_2| + |w(t_1)| + |w(t_2)|$. Thus, $\mathcal{E}_2(t) = L_{s_1}t + 0.2$.

satisfies the constraints of the Assumptions 5.3.1 and 5.3.2. Using Algorithm 1, we compute a robustness estimate of $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \rrbracket_D(\mu_1) = 0.7428$ where $\mu_1 = (\sigma_1, \hat{\tau}_1)$, while $\mathcal{E}_1(\Delta\hat{\tau}_1) = 0.6$. Therefore, by Corollary 5.3.1 we conclude that $\llbracket \phi_1 \rrbracket_C(s_1) = \llbracket \phi_1 \rrbracket_D(\mu_1) = \top$.

The next example manifests a very intuitive attribute of the framework, namely, that the more robust a signal is with respect to the MTL specification the larger the sampling period can be.

Example 5.5.2. Consider the discrete-time signal σ_2 in Fig. 5.1. The MITL specification is $\phi_2 = \square_{[0,4\pi]}p_{21} \wedge \diamond_{[3\pi,4\pi]}p_{22}$ with $\mathcal{O}(p_{21}) = [-4, 4]$ and $\mathcal{O}(p_{22}) = \mathbb{R}_{\leq 0}$. In this case, we compute a robustness estimate of $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu_2) = 1.7372$ where $\mu_2 = (\sigma_2, \hat{\tau}_2)$, while $\mathcal{E}_2(\Delta\hat{\tau}_2) = 1.7$ where $\Delta\hat{\tau}_2 = 0.5$. Therefore, by Corollary 5.3.1 we conclude that $\llbracket \phi_2 \rrbracket_C(s_2) = \top$.

In the following example, we utilize our framework in order to test trajectories of nonlinear systems. More specifically, we consider linear feedback systems with saturation. Such systems have nonlinearities that model sensor/actuator constraints (for example see [111, §10]).

Example 5.5.3 (Example 10.5 in [111]). Consider the following linear dynamical system with nonlinear feedback

$$\dot{x}(t) = Ax(t) - b \text{sat}(cx(t)), \quad s_3(t) = cx(t) \quad (5.9)$$

where the saturation function sat is defined as

$$\text{sat}(y) = \begin{cases} -1 & \text{for } y < -1 \\ y & \text{for } |y| \leq 1 \\ 1 & \text{for } y > 1 \end{cases}$$

and A , b , c are the matrices

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 2 & 1 \end{bmatrix}.$$

First note that the origin $x = [0 \ 0]^T$ is an equilibrium point of the system and that the system is absolutely stable with a finite domain (also note that A is not Hurwitz). An estimate of the region of attraction of the origin is the set $\Omega = \{x \in \mathbb{R}^2 \mid V(x) \leq 0.34\}$, where $V(x) = x^T P x$ and

$$P = \begin{bmatrix} 0.4946 & 0.4834 \\ 0.4834 & 1.0774 \end{bmatrix}$$

(see Example 10.5 in [111] for details). For any initial condition $x(0) \in \Omega$, we know that $x(t) \in \{x \in \mathbb{R}^2 \mid V(x) \leq V(x(0))\}$ for all $t \in \mathbb{R}$. In addition, the distance of $x(t)$ from the origin $[0 \ 0]^T$ is always bounded by the radius of the minimum ball that contains the ellipsoid $\{x \in \mathbb{R}^2 \mid V(x) \leq V(x(0))\}$. The lengths of the axis of the ellipsoid are given by the square roots of the eigenvalues of the matrix

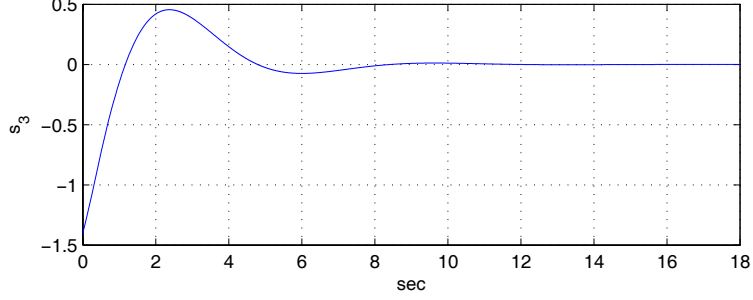


Figure 5.2: The output signal s_3 of Example 5.5.3.

$P_e = V(x(0))P^{-1}$ (see §2.2.2 in [26]). Let $\lambda_{\max}(P_e)$ be the maximum eigenvalue of P_e , then $\|x(t)\| \leq \sqrt{\lambda_{\max}(P_e)}$ for all $t \in R$ and

$$\|\dot{x}(t)\| \leq \|A\|\|x(t)\| + \|b\| \leq \|A\|\sqrt{\lambda_{\max}(P_e)} + \|b\| = L_x$$

Thus, for any $t, t' \in R$, we have

$$|s_3(t) - s_3(t')| \leq \|c\|\|x(t) - x(t')\| \leq \|c\|L_x|t - t'|.$$

That is, $\mathcal{E}_3(t) = \|c\|L_x t$. Assume, now, that we would like to verify that the signal enters an acceptable stability region within 6 to 8sec, that is, the MITL formula is $\phi_3 = \diamond_{[6,8]}\square_{[0,10]}p_{31}$ with $\mathcal{O}(p_{31}) = [-0.25, 0.25]$. The initial condition is $x(0) = [-1 \ 0.6]^T \in \Omega$. The system (5.9) is integrated with a maximum step-size of 0.001 using the MATLAB ode45 solver. The observable discrete-time signal σ_3 has maximum step-size $\Delta\hat{\tau}_3 = 0.045$. The robustness estimate is $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_3) \rrbracket_D(\mu_3) = 0.2372$ where $\mu_3 = (\sigma_3, \hat{\tau}_3)$, while $\mathcal{E}_3(\Delta\hat{\tau}_3) = 0.2182$. Therefore, by Corollary 5.3.1 we conclude that $\llbracket \phi_3 \rrbracket_C(s_3) = \top$. Note that in this example, we assume that the simulation is accurate and, hence, we ignore the possible simulation error

5.6 Related Research

We should point out that the idea of continuous-time temporal logic verification by discrete-time methods is not new. In [97], the relationship between analog and digital clocks for timed state sequences is studied. In this paper, the authors demonstrate that discrete-time verification techniques can be applied to the verification of bounded time invariance and bounded time response properties of continuous-time systems that can be modeled by timed transition systems. A more generalized version of the same problem is studied in [164]. In [52], the authors show that if a formula has the finite variability property, then its validity in discrete time implies validity in continuous time. This result enables the application of verification rules for discrete-time semantics to continuous-time problems.

The work that is the most related to ours appears in [71]. There, the authors give conditions that enable the uniform treatment of both discrete and continuous-time semantics within the temporal logic TRIO (they also note that their results should be easily transferable to MTL). Despite the apparent differences (for example, we do not assume finite variability and we use analog clocks in our discrete-time logic) between [71] and our work, the two approaches are in fact complementary. We actually provide concrete and practical conditions on the signals such that what is defined as “closure under inverse sampling” in [71] holds.

5.7 Conclusions and Future Work

The main contribution of this chapter is a framework that enables continuous-time reasoning using discrete-time methods. In particular, we have achieved two goals. First, we can infer the continuous-time satisfiability of an MITL formula with respect to a continuous-time signal. Our solution utilizes the notion of discrete-time robust-

ness of MTL specifications and provides conditions on the signal dynamics and the sampling function which, by the way, is not required to have a constant sampling rate. Second, we can compute bounds on the continuous-time robustness estimate of an MTL formula – which contains only closed intervals as timing constraints – with respect to a continuous-time signal. In this case, it is required that the sampling function has a constant sampling rate. The second contribution is quite interesting since it might be the only way to under-approximate the continuous-time robustness degree of a temporal logic formula with respect to a continuous-time signal.

In the front of continuous-time verification by discrete-time reasoning, there exist several directions for future research. In the current framework, we require a global bound $\mathcal{E}(\Delta\hat{\tau})$ on the deviation of the signal between two samples. This might be too conservative for applications with variable sampling step. One important modification to this theory will be to use local bounds $\mathcal{E}(\hat{\tau}(i) - \hat{\tau}(i - 1))$ in coordination with an on-line monitoring algorithm. Related to the previous modification is the extension of the present methodology to hybrid systems [107]. Currently, hybrid systems can be handled by taking as bound \mathcal{E} the most conservative bound \mathcal{E}_c of all control locations c of the hybrid automaton. Finally, as it is well known, the Lipschitz constant might be a very conservative estimate on the deviation of the signal between two points in time. In future work, we plan to use approximate metrics [82] in order to obtain better bounds.

Chapter 6

Dynamical System Verification using Robust Simulations

6.1 Introduction and Problem Formulation

In this chapter, we consider the problem of MTL verification of dynamical systems. In particular, we are interested in studying the transient behavior of autonomous continuous-time Linear Parameter Varying (LPV) systems. This is an important problem since it cannot be addressed by the classical control theory or other analytical methods. An important characteristic of LPV systems [11] is that they can also capture certain classes of Linear Time Varying (LTV) systems and, even, nonlinear systems. One additional advantage that MTL verification provides is that we can analyze real-time properties of such systems, which was not possible before.

The class of LPV systems is still too general to be automatically verified. Thus, we will only consider systems that satisfy the following additional constraints.

Definition 6.1.1 (Regular LPV Systems). *A regular LPV system $\overline{\Sigma}_*^{\text{LPV}}$ is an autonomous continuous-time LPV system $\overline{\Sigma}_*^{\text{LPV}} = (R, X, X^0, Y, P, A(p), C)$, which sat-*

satisfies the following constraints:

- the time domain R is bounded,
- the set of initial conditions X^0 is an n -dimensional hyper-rectangle,
- the parameter set P is a q -dimensional hyper-rectangle,
- the parameter function $p : R \rightarrow P$ is a piecewise continuous function,
- the matrix $A : P \rightarrow \mathbb{R}^{n \times n}$ is a multi-affine function of the components of p ,

$$A(p(t)) = \sum_{i_1=0}^1 \sum_{i_2=0}^1 \cdots \sum_{i_q=0}^1 A_{i_1, i_2, \dots, i_q} (p^{(1)}(t))^{i_1} (p^{(2)}(t))^{i_2} \cdots (p^{(q)}(t))^{i_q} \quad (6.1)$$

where $p^{(i)}(t)$ denotes the i -th component of the vector function p and $A_{i_1, i_2, \dots, i_q} \in \mathbb{R}^{n \times n}$ are constant matrices.

The constraints that X^0 and P must be hyper-rectangles are important, since they will enable the development of an automatic MTL verification framework.

Definition 6.1.2 (Hyper-rectangle). *A set P is a q -dimensional hyper-rectangle if it can be written as*

$$P := [\underline{p}_1, \bar{p}_1] \times [\underline{p}_2, \bar{p}_2] \times \cdots \times [\underline{p}_q, \bar{p}_q] \quad (6.2)$$

where for $i = 1, \dots, q$, we have $\underline{p}_i, \bar{p}_i \in \mathbb{R}$ and, also, $\underline{p}_i \leq \bar{p}_i$.

Formally, we address the following problem in this chapter.

Problem 6.1.1. *Given an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$ and a continuous-time regular LPV system $\bar{\Sigma}_{\star}^{\text{LPV}}$, verify that $\mathcal{L}(\bar{\Sigma}_{\star}^{\text{LPV}}) \subseteq \mathcal{L}(\phi, \mathcal{O})$.*

The difficulty in solving Problem 6.1.1 is that there exists an uncountably infinite number of traces y of $\overline{\Sigma}_\star^{\text{LPV}}$ to be checked. This is due to the fact that if both the sets X^0 and P are not singletons, then they contain an uncountably infinite number of points. Thus, the verification of $\overline{\Sigma}_\star^{\text{LPV}}$ cannot be done by a finite number of simulations. In the following, we show that using the robust semantics of MTL (Section 3.2) and the notion of bisimulation function (see Section 4.2 or [82]), the bounded time verification of $\overline{\Sigma}_\star^{\text{LPV}}$ is possible by using only a finite number of simulations.

The first step in our approach is the reduction of the regular LPV system into a Linear Time Invariant (LTI) system. In detail, we will compute a δ -approximate bisimulation relation between the regular LPV system and an LTI system. The next step is to verify that the resulting LTI system satisfies the specification with robustness degree larger than δ . Thus, in order to solve Problem 6.1.1, we will also need to address the following problem.

Problem 6.1.2. *Given an MTL formula $\phi \in \text{MTL}_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$, an autonomous continuous-time LTI system $\overline{\Sigma}^{\text{LTI}} = (R, X, X^0, Y, A, C)$, where R is bounded and X^0 is a n -dimensional hyper-rectangle, and a number $\delta > 0$, verify that $E_\rho(\mathcal{L}(\overline{\Sigma}^{\text{LTI}}), \delta) \subseteq \mathcal{L}(\phi, \mathcal{O})$.*

As we mentioned above our method is fundamentally based on numerical simulation techniques [160]. Numerical simulation methods approximate the differential equations of the system $\overline{\Sigma}^{\text{LTI}}$ by algebraic equations which depend on the size of the integration (time) step. Inevitably, there exists some error between the continuous solution of the differential equations and the result of the numerical simulation, which can be driven arbitrarily close to zero [160]. Therefore, we safely ignore this issue by making the following assumption in order to facilitate the presentation of the contributions in this chapter.

Assumption 6.1.1. Consider an MTL formula $\phi \in MTL_{\mathbb{B}}(AP)$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(Y))$ and a continuous-time system $\bar{\Sigma}^{\text{LTI}}$. Let $x_C \in \Lambda(\bar{\Sigma}^{\text{LTI}})$ and $(x_D, \hat{\tau}) \in \mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, R)$ be the numerical solution of $\bar{\Sigma}^{\text{LTI}}$ such that $x_C(0) = x_D(0)$. Then, we assume that the numerical solution (integration step) is such that $\llbracket \phi, \mathcal{O} \rrbracket_D((y_D, \hat{\tau})) \approx \llbracket \phi, \mathcal{O} \rrbracket_C(y_C)$, where $y_C(t) = Cx_C(t)$ and $y_D(t) = Cx_D(t)$.

The symbol \approx has the meaning of approximately equal. Here, we use it in the sense that for $x, y \in \mathbb{R}$, $x \approx y$ iff $|x - y| = \varepsilon \ll x, y$. In problems where Assumption 6.1.1 might seem too strong, we can removed by incorporating the approximation results from Chapter 5. The next example will the running example with which we are going to demonstrate the different steps of our framework.

Example 6.1.1. Consider the following regular LPV system

$$\bar{\Sigma}_{\star a}^{\text{LPV}} = (R_a, \mathbb{R}^2, X_a^0, \mathbb{R}^2, P_a, A_a(p), C_a)$$

where $R_a = [0, 14] \subseteq \mathbb{R}_{\geq 0}$, $X_a^0 = [0.4, 0.8] \times [-0.3, -0.1]$, $p \in P_a = [0, 1]$, $C_a = \mathbf{I}_2$ and

$$A_a(p) = \begin{bmatrix} 0.05p & -2.5 \\ 0.5 & -1 \end{bmatrix},$$

The MTL specification we would like to verify is

$$\psi_a = \square p_{a1} \wedge \square_{\geq 8} p_{a2}$$

where $\mathcal{O}(p_{a1}) = \mathbb{R} \times [-0.6, 0.6]$ and $\mathcal{O}(p_{a2}) = [-0.4, 0.4] \times [-0.4, 0.4]$. Informally, the specification ψ_a states that the value of $x^{(2)}(t)$ should always be in the set $[-0.6, 0.6]$ and that after 8 time units we should have $x^{(i)} \in [-0.4, 0.4]$ for $i = 1, 2$.

Now consider the nonlinear system of Example 4.1.1. Notice that for any $x \in \mathbb{R}$,

we have $\sin^2(x) \in [0, 1]$. Therefore, the LPV system in this examples actually captures the nonlinear system in Example 4.1.1.

6.2 Approximating LPV Systems

The fundamental difficulty in testing an uncertain linear system $\overline{\Sigma}_*^{\text{LPV}}$ is that we have an uncountable number of parameters to test both in the set of initial conditions X_0 and in the set of unknown parameters P . This problem can be alleviated by employing the notion of approximate bisimilarity which was introduced in Section 4.2. This approach does not only allow us to test LTI systems with uncertain parameters, but also LTV systems whose time varying parameters can be contained in a hyper-rectangle.

Let us assume that we are given an autonomous uncertain linear system $\overline{\Sigma}_*^{\text{LPV}}$ and a constant vector of parameters p_0 in P . This vector could be either the nominal operating point of the system, which is provided along with the model of the system, or we can uniformly sample a point from the set P . Then, all we need to do is to find an approximate bisimulation function \mathcal{F} between the systems $\overline{\Sigma}_*^{\text{LPV}}$ and $\overline{\Sigma}_*^{\text{LPV}}(p_0)$. If we find one, and the game (4.3) returns a δ which has a finite value, then we know that $\overline{\Sigma}_*^{\text{LPV}}(p_0)$ can approximate $\overline{\Sigma}_*^{\text{LPV}}$ with precision δ and vice versa. Therefore, we will have essentially reduced the verification problem for $\overline{\Sigma}_*^{\text{LPV}}$ to robustly testing $\overline{\Sigma}_*^{\text{LPV}}(p_0)$.

The next Theorem indicates that an approximate bisimulation function between two LPV systems can be efficiently computed despite the fact we have an uncountable number of parameters. The notation $\mathcal{EP}(P)$ in the following theorem denotes the

extreme points of a convex polytope P , that is,

$$\mathcal{EP}(P) = \{p_0 \in P \mid p_0 = \lambda p_1 + (1 - \lambda)p_2, p_1, p_2 \in P, \lambda \in (0, 1) \implies p_0 = p_1 = p_2\}.$$

When P is a q -dimensional hyper rectangle, then the set of extreme points is simply

$$\mathcal{EP}(P) = \{(w_1, w_2, \dots, w_q) \mid w_i = \underline{p}^i \text{ or } w_i = \bar{p}^i \text{ for all } i = 1, 2, \dots, q\}.$$

Theorem 6.2.1. *Consider two regular LPV systems $\bar{\Sigma}_{\star 1}^{\text{LPV}}$ and $\bar{\Sigma}_{\star 2}^{\text{LPV}}$. If there exists a positive semidefinite matrix M such that*

$$M \geq C_{\star} \tag{6.3}$$

$$\forall p_1 \in \mathcal{EP}(P_1) . \forall p_2 \in \mathcal{EP}(P_2) . A_{\star}^T(p_1, p_2)M + MA_{\star}(p_1, p_2) \leq 0, \tag{6.4}$$

where

$$C_{\star} = \begin{bmatrix} C_1^T C_1 & -C_1^T C_2 \\ -C_2^T C_1 & C_2^T C_2 \end{bmatrix} \text{ and } A_{\star}(p_1, p_2) = \begin{bmatrix} A_1(p_1) & 0 \\ 0 & A_2(p_2) \end{bmatrix},$$

then the function $\mathcal{F}(x) = \sqrt{x^T M x}$ where $x = \begin{bmatrix} x_1^T & x_2^T \end{bmatrix}^T$ is a bisimulation function between $\bar{\Sigma}_{\star 1}^{\text{LPV}}$ and $\bar{\Sigma}_{\star 2}^{\text{LPV}}$.

The following corollary of Theorem 6.2.1 indicates how we can compute a bisimulation function \mathcal{F} between $\bar{\Sigma}_{\star}^{\text{LPV}}$ and $\bar{\Sigma}_{\star}^{\text{LPV}}(p_0)$.

Corollary 6.2.1. *Consider a regular LPV system $\bar{\Sigma}_{\star}^{\text{LPV}}$ and a vector $p_0 \in P$. If there exists a positive semidefinite matrix M such that*

$$M \geq C_{\star} \tag{6.5}$$

$$\forall p \in \mathcal{EP}(P) . A_{\star}^T(p)M + MA_{\star}(p) \leq 0, \tag{6.6}$$

where

$$C_\star = \begin{bmatrix} C^T C & -C^T C \\ -C^T C & C^T C \end{bmatrix} \quad \text{and} \quad A_\star(p) = \begin{bmatrix} A(p_0) & 0 \\ 0 & A(p) \end{bmatrix},$$

then the function $\mathcal{F}(x) = \sqrt{x^T M x}$ where $x = \begin{bmatrix} x_i^T & x_u^T \end{bmatrix}^T$ is a bisimulation function between $\bar{\Sigma}_i = \bar{\Sigma}_\star^{\text{LPV}}(p_0)$ and $\bar{\Sigma}_u = \bar{\Sigma}_\star^{\text{LPV}}$.

The system of equations (6.3) and (6.4) can be efficiently solved with a semidefinite programming solver. Notice, however, that the number of equations in (6.4) increases exponentially with the number of unknown parameters. Theorem 6.2.1 forms the basis for a testing framework for uncertain and time varying linear systems. If an approximate bisimulation function exists between $\bar{\Sigma}_\star^{\text{LPV}}$ and $\bar{\Sigma}_\star^{\text{LPV}}(p_0)$, then the next step is to compute how well $\bar{\Sigma}_\star^{\text{LPV}}(p_0)$ approximates $\bar{\Sigma}_\star^{\text{LPV}}$ by solving the static games in (4.3). As the next proposition establishes, the sup-inf optimizations in (4.3) can be reduced into a number of quadratic programs.

Proposition 6.2.1. *Consider a system $\bar{\Sigma}_\star^{\text{LPV}}$ and a vector $p_0 \in P$ and let*

$$\mathcal{F}(x_i, x_u) = \sqrt{\mathcal{V}(x_i, x_u)} = \sqrt{\begin{bmatrix} x_i^T & x_u^T \end{bmatrix} M \begin{bmatrix} x_i^T & x_u^T \end{bmatrix}^T}$$

be a bisimulation function between $\bar{\Sigma}_i = \bar{\Sigma}_\star^{\text{LPV}}(p_0)$ and $\bar{\Sigma}_u = \bar{\Sigma}_\star^{\text{LPV}}$. Then, the solution of the static games

$$\max\left\{ \max_{x_i^0 \in \mathcal{EP}(X_0)} \inf_{x_u^0 \in X_0} \mathcal{V}(x_i^0, x_u^0), \max_{x_u^0 \in \mathcal{EP}(X_0)} \inf_{x_i^0 \in X_0} \mathcal{V}(x_i^0, x_u^0) \right\} \quad (6.7)$$

computes the optimal points x_i^* and x_u^* which provide an upper bound $\delta \geq \delta^* = \mathcal{F}(x_i^*, x_u^*)$ for the game of eq. (4.3).

Example 6.2.1. *Consider the regular LPV system $\bar{\Sigma}_{\star a}^{\text{LPV}}$ of Example 6.1.1 and the*

derived LTI system $\overline{\Sigma}_{*a}^{\text{LPV}}(p_0)$, where $p_0 = 0.5$. Using Corollary 6.2.1, we compute the following positive semidefinite matrix:

$$M = \begin{bmatrix} 1.1016 & -0.4013 & -1.0903 & 0.3890 \\ -0.4013 & 2.6660 & 0.4021 & -2.6438 \\ -1.0903 & 0.4021 & 1.1105 & -0.4252 \\ 0.3890 & -2.6438 & -0.4252 & 2.7105 \end{bmatrix}$$

Using Proposition 6.2.1, the approximation bound δ_a between $\overline{\Sigma}_{*a}^{\text{LPV}}$ and $\overline{\Sigma}_{*a}^{\text{LTI}}(p_0)$ is computed to be 0.2125. The whole process took 0.93sec to compute using MATLABTM on PIII mobile 1.2GHz with 1GB of RAM.

6.3 MTL Robust Testing of LTI Systems

In this section, we show that Problem 6.1.2 can be solved in the framework of continuous-time dynamical systems. Our approach comprises three basic steps. First, we define a notion of neighborhood on the set of trajectories of the system $\overline{\Sigma}^{\text{LTI}}$. This enables us to determine the sets of trajectories with approximately equivalent behaviors. Then, it is possible to verify that a property ϕ holds for all the traces of the dynamical system by simulating only a finite number of traces of the system $\overline{\Sigma}^{\text{LTI}}$ and evaluating their coefficients of robustness.

6.3.1 Sampling Using Bisimulation Functions

The next result states that by using the robust semantics of MTL and a bisimulation function, it is possible to infer whether the robustness degree of an infinite number of traces with respect to the specification is at least δ .

Theorem 6.3.1. Let $\bar{\Sigma}^{\text{LTI}}$ be a continuous-time LTI system, $\phi \in \text{MTL}_{\mathbb{B}}(\text{AP})$ be an MTL formula, $\mathcal{O} \in \mathfrak{F}(\text{AP}, \mathcal{P}(Y))$ be an observation map, \mathcal{F} be a bisimulation function of $\bar{\Sigma}^{\text{LTI}}$ with itself, $x_1 \in \Lambda(\bar{\Sigma}^{\text{LTI}})$ be a state trajectory of $\bar{\Sigma}^{\text{LTI}}$ and $y_1(t) = Cx_1(t)$. Consider any number $\delta \geq 0$ and any $x_2 \in \Lambda(\bar{\Sigma}^{\text{LTI}})$ such that $0 < \mathcal{F}(x_1(0), x_2(0)) < \llbracket \phi, \mathcal{O} \rrbracket_C(y_1) - \delta$, then $\text{Dist}_{\rho}(y_2, \mathcal{L}(\phi, \mathcal{O})) \geq \delta$, where $y_2(t) = Cx_2(t)$.

The challenge in developing a simulation-based verification algorithm is to sample the set of initial conditions in a way that ensures coverage. For this purpose, we define a discretization operator based on the bisimulation function.

Proposition 6.3.1. Let $\mathcal{F} : X \times X \rightarrow \bar{\mathbb{R}}_{\geq 0}$ be a bisimulation function. For any compact set of initial conditions $X^0 \subseteq X$, for all $\varepsilon > 0$, there exists a finite set of points $\{x_1, \dots, x_k\} \subseteq X^0$ such that

$$\text{for all } x \in X^0, \text{ there exists } x_i, \text{ such that } \mathcal{F}(x, x_i) \leq \varepsilon. \quad (6.8)$$

Let Disc be the discretization operator which maps the compact set $X^0 \subseteq X$ and a strictly positive number ε to a list of points $\text{Disc}(X^0, \varepsilon) = \{x_1, \dots, x_r\}$ satisfying equation (6.8).

Theorem 6.3.2. Let $x_1, \dots, x_k \in \Lambda(\bar{\Sigma}^{\text{LTI}})$ be state trajectories of $\bar{\Sigma}^{\text{LTI}}$ such that

$$\text{Disc}(X^0, \varepsilon) = \{x_1(0), \dots, x_k(0)\}$$

and y_1, \dots, y_k be the associated observation trajectories, i.e. $y_i(t) = Cx_i(t)$, then for any $\delta \geq 0$, we have

$$(\forall i = 1, \dots, k. \llbracket \phi, \mathcal{O} \rrbracket_C(y_i) > \varepsilon + \delta) \implies (E_{\rho}(\mathcal{L}(\bar{\Sigma}^{\text{LTI}}), \delta) \subseteq \mathcal{L}(\phi, \mathcal{O}))$$

Thus, it is possible to verify that the system $\bar{\Sigma}^{\text{LTI}}$ satisfies the MTL property ϕ with robustness at least δ by evaluating the robustness estimate of only a finite number of state trajectories.

6.3.2 Verification Algorithm

Algorithm 3 verifies that the property ϕ holds with at least robustness degree δ on $\bar{\Sigma}^{\text{LTI}}$. The main idea is the following. We start with a rough discretization (using a parameter $\varepsilon > 0$) of the set of initial states – typically we pick just one point. Then, we try to verify the property using the traces associated with these initial states. When the result of the verification is inconclusive (for example when $\delta \leq \llbracket \phi, \mathcal{O} \rrbracket_D(\mu) \leq \delta + \varepsilon$ for simulated trajectory μ), the discretization of the initial states is refined locally (using a refinement parameter $r \in (0, 1)$) around the initial states for which we were unable to conclude the property. This algorithm, therefore, allows the fast verification of robust properties, whereas more computational effort is required for non-robust properties. The refinement operation is repeated at most K times (a user defined parameter). The algorithm can terminate in one of four possible states:

1. the property holds with at least robustness degree δ on the system $\bar{\Sigma}^{\text{LTI}}$,
2. the property has been falsified – we have found a trace that does not satisfy the specification, i.e., negative robustness estimate,
3. the system $\bar{\Sigma}^{\text{LTI}}$ is not δ robust with respect to the property, or
4. we have computed a subset \check{X}^0 of the initial states X^0 such that all the state trajectories initiating from \check{X}^0 satisfy the MTL property with robustness degree at least δ .

In the last case, we also get a degree of coverage of the initial states that have been verified. The proof of the correctness of the algorithm is not stated here but is very similar to that of Theorem 6.3.2. Note, however, that the correctness of this verification framework depends critically on Assumption 6.1.1.

Algorithm 3 Temporal Logic Verification Using Simulation

Require: A system $\overline{\Sigma}^{\text{LTI}} = (R, X, X^0, Y, A, C)$, an MTL formula ϕ , an observation map \mathcal{O} and numbers $\varepsilon > 0$, $\delta \geq 0$, $r \in (0, 1)$, $K \in \mathbb{N}$.

```

1: procedure VERIFY( $\overline{\Sigma}^{\text{LTI}}$ ,  $\phi$ ,  $\mathcal{O}$ ,  $\varepsilon$ ,  $\delta$ ,  $r$ ,  $K$ )
2:    $\hat{X}^0 \leftarrow \text{Disc}(X^0, \varepsilon)$ ,  $\check{X}^0 \leftarrow \emptyset$ ,  $k \leftarrow 0$ 
3:   while  $k \leq K$  and  $\hat{X}^0 \neq \emptyset$  do
4:      $\hat{X}_{tmp}^0 \leftarrow \emptyset$ 
5:     for  $x^0 \in \hat{X}^0$  do
6:        $\mu \leftarrow \text{Simulate } \Sigma \text{ for the time in } R \text{ from initial state } x^0$ 
7:       if  $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu) < 0$  then
8:         return “ $\phi$  does not hold on  $\overline{\Sigma}^{\text{LTI}}$ ,”
9:       else if  $0 \leq \llbracket \phi, \mathcal{O} \rrbracket_D(\mu) < \delta$  then
10:        return “ $\overline{\Sigma}^{\text{LTI}}$  is not  $\delta$ -robust wrt to  $\phi$ ”
11:       else if  $\delta \leq \llbracket \phi, \mathcal{O} \rrbracket_D(\mu) < \delta + r^k \varepsilon$  then
12:         $\hat{X}_{tmp}^0 \leftarrow \hat{X}_{tmp}^0 \cup \text{Disc}(X^0 \cap \mathcal{N}_{\mathcal{F}}(x^0, r^k \varepsilon), r^{k+1} \varepsilon)$ 
13:       else
14:         $\check{X}^0 \leftarrow \check{X}^0 \cup \mathcal{N}_{\mathcal{F}}(x^0, r^k \varepsilon)$ 
15:       end if
16:     end for
17:      $k \leftarrow k + 1$ ,  $\hat{X}^0 \leftarrow \hat{X}_{tmp}^0$ 
18:   end while
19:   if  $\hat{X}_{tmp}^0 = \emptyset$  then
20:     return “ $\phi$  holds on  $\overline{\Sigma}^{\text{LTI}}$  with robustness degree at least  $\delta$ ”
21:   else
22:     return “ $\phi$  holds  $\delta$ -robustly on  $\overline{\Sigma}^{\text{LTI}} = (R, X, X^0 \cap \check{X}^0, Y, A, C)$ ”
23:   end if
24: end procedure
25: ▷ In lines 14,12:  $\mathcal{N}_{\mathcal{F}}(x, \varepsilon) = \{x' \in X \mid \mathcal{F}(x, x') \leq \varepsilon\}$ 

```

Remark 6.3.1. Consider replacing $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu)$ in Algorithm 3 by the theoretical quantity $\text{Dist}_{\hat{\rho}}(\mu^{(1)}, \mathcal{L}^{\hat{\tau}}(\phi, \mathcal{O}))$ for some sampling function $\hat{\tau} \in \mathfrak{F}^{\uparrow}(N, R)$. In this case, it can be shown that whenever $\text{dist}_{\hat{\rho}}(\mathcal{L}^{\hat{\tau}}(\overline{\Sigma}^{\text{LTI}}), \mathcal{L}^{\hat{\tau}}(\neg\phi, \mathcal{O})) > 0$, the algorithm is com-

plete and can verify the system using only a finite number of simulations. The current algorithm may fail to be complete since we are using an under-approximation of the robustness degree (note also that $\llbracket \phi, \mathcal{O} \rrbracket_D(\mu) = 0 \not\Rightarrow \mathbf{Dist}_{\hat{\rho}}(\mu^{(1)}, \mathcal{L}^{\hat{r}}(\phi, \mathcal{O})) = 0$).

The next example demonstrates how our verification toolbox in MATLABTM works for testing autonomous LTI systems.

Example 6.3.1. *Let us go back to the LTI system $\bar{\Sigma}_a^{\text{LTI}} = \bar{\Sigma}_{x_a}^{\text{LPV}}(p_0)$ of Example 6.2.1. We have to prove that $\bar{\Sigma}_a^{\text{LTI}}$ satisfies the specification ψ_a of Example 6.1.1 with robustness degree at least $\delta_a = 0.2125$ (see Example 6.2.1).*

At the initialization step, the algorithm computes a bisimulation function $\mathcal{F}'(x) = \sqrt{x^T M' x}$ of $\bar{\Sigma}^{\text{LTI}}$ with itself by solving the following set of matrix equations

$$\begin{aligned} M &\geq C_a^T C_a = \mathbf{I}_2 \\ A_a^T(p_0)M' + M'A_a(p_0) &\leq 0, \end{aligned}$$

The positive semidefinite matrix M' that defines the bisimulation function is computed using SeDuMi [170] to be:

$$M' = \begin{bmatrix} 1.0940 & -0.3895 \\ -0.3895 & 2.6142 \end{bmatrix}$$

The next step in the procedure is to pick a point x^0 from the set of initial conditions X_a^0 for the first simulation. This is always chosen to be the centroid of the set of initial conditions, e.g., $x^0 = [0.6 \ -0.2]^T$ in this case. Then, we must compute the maximum value ε of the bisimulation function over the set of initial conditions relative to the point x^0 , i.e., $\varepsilon = \sup_{x \in X_a^0} \mathcal{F}'(x - x^0)$. Since X_a^0 is a hyper-rectangle and \mathcal{F}' is a convex function, the maximum value is attained on one of the extreme points

of X_a^0 . In other words, $\varepsilon = \max_{x \in \mathcal{EP}(X_a^0)} \mathcal{F}'(x - x^0)$. For this example, we compute $\varepsilon = 0.2924$.

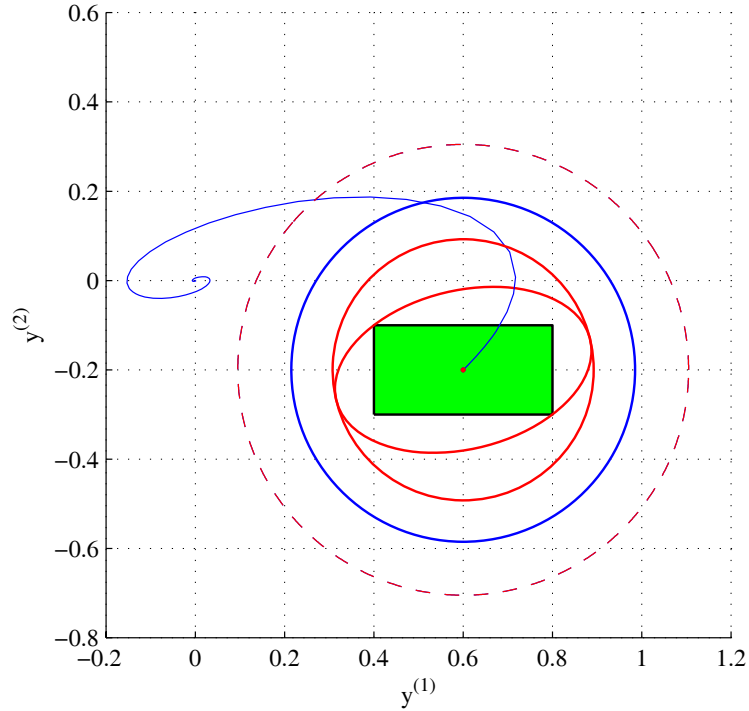


Figure 6.1: 1st iteration of the algorithm. The green rectangle indicates the set of initial conditions in the output space. The ellipsoid indicates the level set of the bisimulation function for $\varepsilon = 0.2924$ and the enclosing circle has radius $\varepsilon = 0.2924$. The plotted trajectory has robustness estimate 0.3852 (blue circle), while $\delta_a + \varepsilon = 0.5049$ (dashed red circle).

Figure 6.1 summarizes the first iteration of the algorithm. Let $\mu_0 = (y_0, \hat{\tau}_0)$ denote the timed state sequence defined by pairing the observation trajectory y_0 with the sampling function τ_0 which are the outcome of the simulation of $\bar{\Sigma}_a^{\text{LTI}}$ for 14 time units and with initial conditions $x^0 = [0.6 \ -0.2]^T$. The robustness estimate of μ_0 is $\llbracket \psi, \mathcal{O} \rrbracket_D(\mu_0) = 0.3852$, while $\delta_a + \varepsilon = 0.5049$. Therefore, the algorithm refines locally the points which must be tested. For the refinement process, we always pick the centroids of the hyper-rectangles that are generated by dividing the initial hyper-rectangle with respect to the testing point $-x^0$ in this iteration (see Figure 6.2). By

choosing $r = 0.5$, we can guarantee that the value of the bisimulation function on all the points in the new hyper-rectangles relative to their respective centroids is bounded by $\varepsilon/2$.

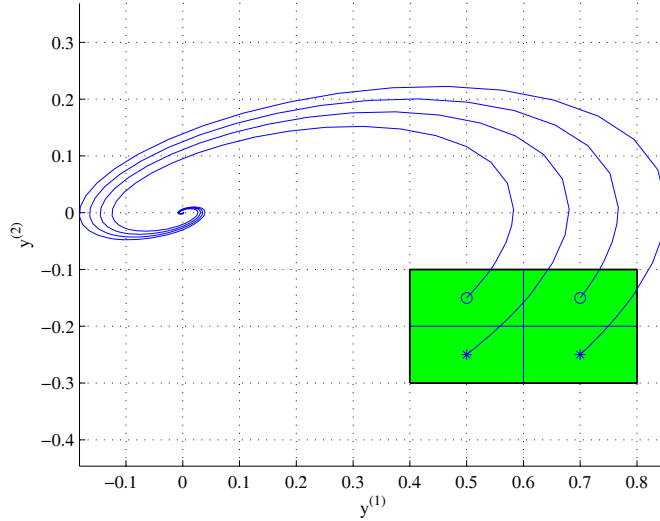


Figure 6.2: 2nd iteration of the algorithm. The new hyper-rectangles and their respective centroids that result from the refinement process of X_a^0 with respect to x^0 . The circles indicate testing points that initiate trajectories that satisfy ψ_a δ_a -robustly, while the stars indicate testing points that must be refined locally.

The process repeats until the algorithm terminates in one of the states described earlier in this section. In this example, the algorithm found $\overline{\Sigma}_a^{\text{LTI}}$ to be δ_a -robust with respect to ψ_a . The verification process required 13 simulations in total and it took 6.5 sec to complete. The 10 simulations that prove the correctness (robustness) of the system appear in Figures 6.3 and 6.4.

The next example demonstrates that the proposed algorithm can verify systems with large state spaces (many continuous variables).

Example 6.3.2. Here, we present a transmission line example which we borrowed from [88]. The goal is to check whether the transient behavior of a long transmission line is acceptable both in terms of overshoot and response time.

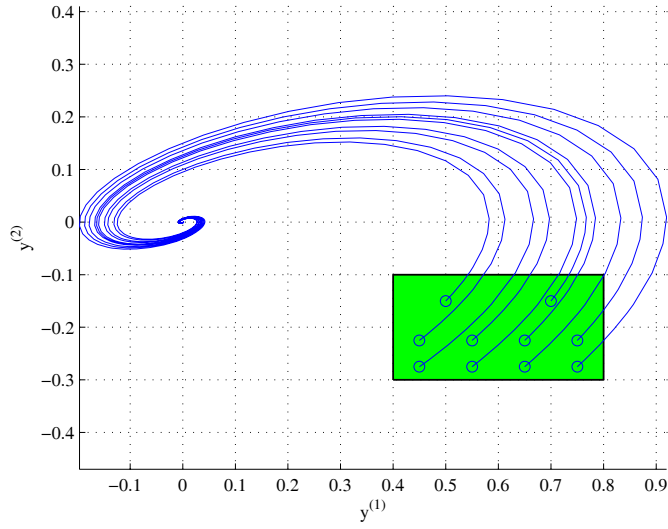


Figure 6.3: The 10 simulations that verify that the system is at least 0.2125-robust with respect to specification ψ_a : Phase space.

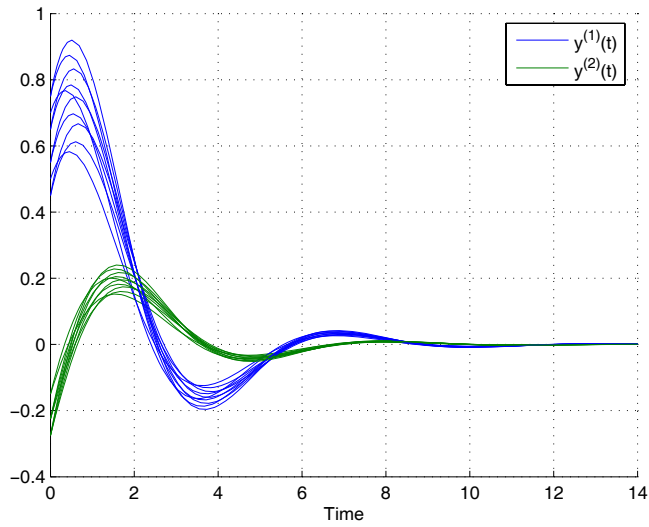


Figure 6.4: The 10 simulations that verify that the system is at least 0.2125-robust with respect to specification ψ_a : Time domain.

As mentioned in Example 4.1.2, a transmission line can be modeled by an LTI system. Let $\Sigma_c^{\text{LTI}} = (R_c, \mathbb{R}^{81}, X_c^0, \mathbb{R}, \mathbb{R}, A_c, b_c, C_c)$ be the LTI system, where $R_c = [0, 2]$ and $X_c^0 = \{-A^{-1}bu \mid u \in [-0.2, 0.2]\}$. The matrices A_c , b_c and C_c are too large to be explicitly presented here, but further details can be found in [88]. Note that the system we are trying to verify is 81-dimensional. Initially, $u(0) \in U_c^0 = [-0.2, 0.2]$ and the system is at its steady state $x(0) = -A_c^{-1}b_c u(0)$. Then, at time $t > 0$ the input is set to the value $u(t) = 1$. The output of the system (observable trajectory) for $u(0) = 0$ appears in Fig. 6.5.

The goal of the verification is double. We want to check that the voltage at the receiving end stabilizes between 0.8 and 1.2 Volts within T nano-seconds (response time) and that its amplitude always remains bounded by θ Volts (overshoot) where $T \in [0, 2]$ and $\theta \geq 0$ are design parameters. The specification is expressed as the MTL property:

$$\psi_c = \square p_{c1} \wedge \diamond_{[0,T]} \square p_{c2}$$

where the predicates are mapped as follows: $\mathcal{O}(p_{c1}) = [-\theta, \theta]$ and $\mathcal{O}(p_{c2}) = [0.8, 1.2]$.

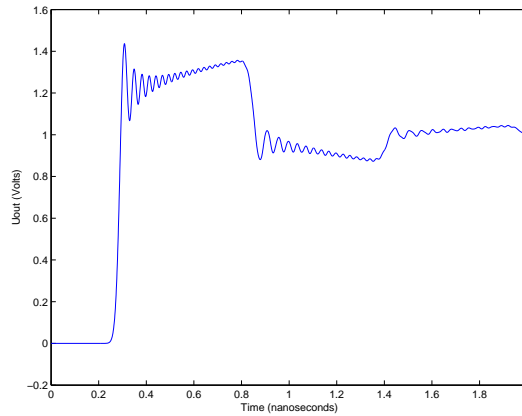


Figure 6.5: An example trace of the RLC model of the transmission line.

We can compute a bisimulation function $\mathcal{F}(x) = \sqrt{x^T M x}$ of the system with itself

by solving the following set of matrix equations

$$M \geq C_c^T C_c$$

$$A_c^T M + M A_c \leq 0,$$

We run the verification algorithm for $T \in \{0.8, 1.2, 1.6\}$, $\theta \in \{1.4, 1.5, 1.6\}$ and robustness parameter $\delta = 0$. The results are summarized in Table 6.1. For the cases where the property ψ_c holds on Σ_c^{LTI} , we can see that the number of simulations needed for the verification is related to the robustness of the system with respect to the property. Indeed, the larger the T and θ are, the more robust the Σ_c^{LTI} is with respect to the property ψ_c and, in turn, the less is the number of the simulations that are required for verification. This is one interesting feature of our approach which relates robustness to the computational complexity of the verification.

	$T = 0.8$	$T = 1.2$	$T = 1.6$
$\theta = 1.4$	False / 1	False / 1	False / 1
$\theta = 1.5$	False / 1	False / 15	False / 13
$\theta = 1.6$	False / 1	True / 17	True / 7

Table 6.1: Experimental results of the verification algorithm for the transmission line example. For each value of (T, θ) the table gives whether the property ψ_c holds on Σ_c^{LTI} and how many simulations of the system were necessary to conclude.

6.4 Putting Everything Together

The previous sections outlined the theoretical results that comprise the basic building blocks for a framework for the MTL testing and verification of LPV systems. This section briefly discusses how these results can be put together into a practical and efficient testing algorithm and it concludes with some numerical results.

Assume that we are given a regular LPV system $\overline{\Sigma}_\star^{\text{LPV}}$ and an MTL formula ϕ (along with the corresponding observation map \mathcal{O}). First, we choose a random vector p_0 from the set parameter values P . If a nominal parameter vector p_0 has been defined, then we use that instead of a random vector. Then using Corollary 6.2.1, we compute a bisimulation function \mathcal{F} between $\overline{\Sigma}_\star^{\text{LPV}}(p_0)$ and $\overline{\Sigma}_\star^{\text{LPV}}$. If such a bisimulation function exists, then the next step is to determine the accuracy δ of the approximate bisimulation relation. This is done using Proposition 6.2.1. Note that all the above steps can be efficiently computed within MATLABTM using the Optimization ToolboxTM and SeDuMi [170]. Finally, the resulting problem $E_\rho(\mathcal{L}(\overline{\Sigma}_\star^{\text{LPV}}(p_0)), \delta) \subseteq \mathcal{L}(\phi, \mathcal{O})$ can be solved using the MTL robust testing algorithm. Next, we present some numerical examples using the prototype MATLAB toolbox that we have developed. All the numerical experiments were performed on a PIII mobile 1.2GHz with 1GB of RAM.

Example 6.4.1. *Consider a modified version of Example 4.1.2 where now the system has also unknown parameters. In detail, assume that the parameters r and l are known and constant with values $r = 2.5$ and $l = 1.25$, while the exact value of the capacitances is uncertain and possibly time varying such that $c(t) \in [c_0 - \beta, c_0 + \beta]$, where $c_0 = 3.75 \cdot 10^{-3}$ and $\beta = 2 \cdot 10^{-5}$. Formally, we are trying to verify the system $\Sigma_b^{\text{LPV}} = (R_b, \mathbb{R}^{10}, X_b^0, \mathbb{R}^5, \mathbb{R}, P_b, A'_b, b_b, C_b)$, where $P_b \in [c_0 - \beta, c_0 + \beta]^5$ and $X_b^0 = \prod_{i=1}^5 (\{0\} \times [-\alpha, \alpha])$. The structure of the matrix $A'_b(p)$ appears in Figure 6.4.1. Note that even though the matrix $A'_b(p)$ is not a multi-affine matrix valued function as required in the definition of a regular LPV system (see Definition 6.1.1), it can be converted into that form by the simple transformation $p' = \frac{1}{c_0+p}$. Thus, if $p \in [\underline{p}, \overline{p}]$, then $p' \in [\frac{1}{c_0+\overline{p}}, \frac{1}{c_0+\underline{p}}]$.*

Another detail we should point out is that Proposition 6.2.1 holds for autonomous linear systems of the form $\dot{x} = A(p)x$ with $p \in P$. However, the closed-loop system $\overline{\Sigma}_b^{\text{LPV}}$ under the step input $u(t) = 1$ for $t \geq 0$ is of the form $\dot{x} = A(p)x + b$. We

$$A'_b(p) = \begin{bmatrix} -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{c_0+p^{(1)}} & 0 & -\frac{1}{c_0+p^{(1)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{c_0+p^{(2)}} & 0 & -\frac{1}{c_0+p^{(2)}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{c_0+p^{(3)}} & 0 & -\frac{1}{c_0+p^{(3)}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{c_0+p^{(4)}} & 0 & -\frac{1}{c_0+p^{(4)}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{l} & -\frac{r}{l} & -\frac{1}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{c_0+p^{(5)}} & 0 \end{bmatrix}$$

Figure 6.6: The matrix $A'_b(p)$ of Example 6.4.1.

can convert the latter system into an equivalent $\dot{z} = A(p)z$ under the transformation $z = x + A^{-1}(p)b$ if $A(p)$ is invertible for all $p \in P$. Since X^0 is a hyper-rectangle, it can be described by a set of inequalities of the form $c_i x \leq d_i$. In this case, the new set of initial conditions Z^0 will be described by the set of inequalities $c_i z \leq d_i + c_i A^{-1}(p)b$. The resulting set for all $p \in P$, i.e., $Z^0 = \cup_{p \in P} \{z \mid \wedge_i c_i z \leq d_i + c_i A^{-1}(p)b\}$, might not be a hyper-rectangle or even a convex set! Nevertheless, in this example it is the case that $b' = A^{-1}(p)b = [0 \ -1 \ 0 \ -1 \ 0 \ -1 \ 0 \ -1 \ 0 \ -1]^T$, that is, b' is independent of the parameter values. Therefore, the set Z^0 is simply a translation of X^0 and, thus, it is still a hyper-rectangle.

Problem Instance	1	2
α	0.03	0.06
δ	0.1150	0.1198

Table 6.2: Approximation bounds for the Problems of Example 6.4.1.

As the nominal parameter value, we pick the vector $p_0 = [0 \ 0 \ 0 \ 0 \ 0]^T$. Table 6.4.1 indicates how the approximation δ between the LPV system and the corresponding LTI system changes with respect to the parameter α , i.e., the size of the set of initial

conditions X_0 . The computation time for the approximation bound δ for this problem size takes about 15.69 sec in MATLAB.

We need to verify that the voltage at all the nodes does not exceed 2 voltage units and that the voltage at the receiving end, i.e., $y^{(5)} = x^{(10)}$, stabilizes within 3 time units within the range $[0.75, 1.25]$. The above requirement is captured by the MTL formula $\psi_b = \square p_{b1} \wedge \diamond_{\leq 2} \square p_{b2}$, where $\mathcal{O}(p_{b1}) = \{y \in \mathbb{R}^5 \mid \bigwedge_{i=1}^5 |y^{(i)}| \leq 2\}$ and $\mathcal{O}(p_{b2}) = \{y \in \mathbb{R}^5 \mid 0.8 \leq y^{(5)} \leq 1.2\}$. Table 6.3 summarizes the verification results. The simulated trajectory for Problem Instance 1 appears in Figure 4.6, while the trajectories that verify the Problem Instance 2 appear in Figure 6.7.

Problem Instance	1	2
safe	✓	✓
time (sec)	3.46	76.27
# simulations	1	33

Table 6.3: Verification results for Example 6.4.1.

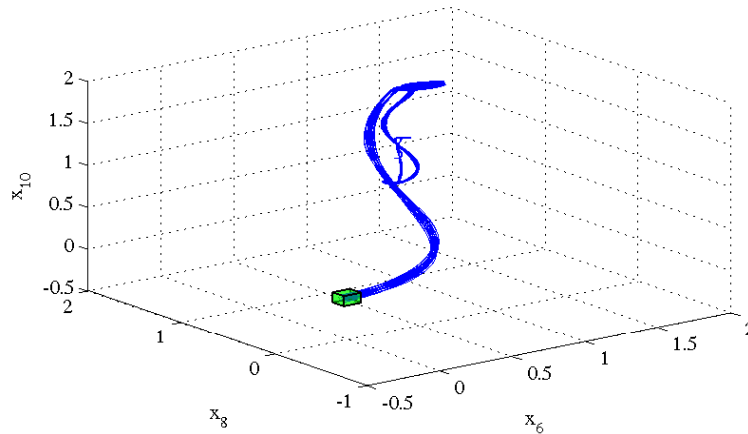


Figure 6.7: The 33 trajectories that are necessary for the verification of Problem Instance 2 of Example 6.4.1. This is the phase space for the variables $y^{(3)} - y^{(4)} - y^{(5)}$. The box indicates the set of initial conditions.

The next example demonstrates how the framework can be used for the real-time verification of nonlinear systems.

Example 6.4.2. *Let's consider again the running example of this chapter. In Example 6.1.1, we indicated that the nonlinear system $\bar{\Sigma}_a$ of Example 4.1.1 can be captured by the LPV system $\bar{\Sigma}_a^{\text{LPV}}$. Then in Examples 6.2.1 and 6.3.1, we showed that the LPV system is correct with respect to the specification ψ_a with robustness degree at least δ_α . Thus, we can conclude that $\bar{\Sigma}_a$ also satisfies the specification ψ_a .*

6.5 Related Research

There is a lot research activity in the verification and testing of dynamical systems. In the introduction, we sampled a few of the works of the vast literature in testing and verification. Here, we will only focus on frameworks that use robust simulations or temporal logics.

Girard and Pappas in [80] have developed a methodology for the verification of safety properties of continuous-time dynamical systems with inputs. In [54], the authors develop a robust testing framework for safety properties using sensitivity analysis. The main differences between [54] and the methodology in this chapter are (i) that we use a more expressive specification language, namely MTL, and (ii) that we use approximate bisimulation relations instead of sensitivity analysis. The authors in [107] have presented a framework for the robust testing of safety properties of hybrid systems.

An application area that has attracted the interest of the verification community is the analysis of analog and mixed-signal circuits. Such systems can be modeled by continuous and hybrid dynamical systems respectively. Since the properties that need to be tested on such systems extend beyond simple safety requirements, there is a lot of interest in introducing temporal logics as specification languages. For example in [90], the authors generate conservative discrete approximations of the continuous

state space of nonlinear systems. Then, the discretized models are verified against specifications expressed in an extension of Computational Tree Logic (CTL) [41]. The paper [47] presents a methodology for generating discretized models both in state space and in time of nonlinear systems. These discrete models are captured by Finite State Machines (FSM). Again, the requirements are expressed in an extension of CTL and a bounded model checking algorithm is applied to the system. In [128], a method for constructing Labeled Hybrid Petri Nets (LHPNs) from simulation traces of the circuit is presented. Similar to our verification framework, this approach can also handle varying parameters. Then, the resulting Petri Net is model checked using a variety of methods proposed by the authors in their previous works. Note that the last two methodologies are falsification rather than verification frameworks due to the way the FSMs or the LHPNs are generated.

6.6 Conclusions and Future Work

In this chapter, we have presented a framework for the Metric Temporal Logic (MTL) testing and verification of Linear Parameter Varying (LPV) systems. This class of systems includes linear systems with uncertain parameters and some instances of linear time varying, nonlinear and hybrid systems.

The main contributions of this chapter are two. First, we present the construction of a bisimulation function that enables the approximation of an LPV system by a Linear Time Invariant (LTI) system in an computationally efficient way. Moreover, the results have been developed in such a way that allow the temporal logic verification step on the LTI system to be performed by any algorithm that can check MTL properties.

However, since – to the best of our knowledge – such a verification algorithm,

which can handle systems with large continuous state spaces, does not exist, we have proposed a framework for the bounded time MTL verification of LTI systems. Note that the framework can be extended to handle other classes of systems, too, as long as we can find bisimulation functions. The proposed methodology reinforces a very intuitive observation : robustly (safe or unsafe) systems are easier to verify. We believe that light weight verification methods, such as the one presented here, can offer valuable assistance to the practitioner.

Future research will concentrate on incorporating the results of this chapter with the works presented in [80] and [107]. This will enable the verification of hybrid systems with unknown parameters under the presence of uncertainty.

Chapter 7

Temporal Logic Motion Planning

7.1 Introduction and Problem Formulation

This chapter deals with the problem of motion generation for mobile robots from high level specifications. We consider a mobile robot which is modeled by the second order system Σ (*dynamics model*):

$$\ddot{x}(t) = u(t), \quad x(t) \in X, x(0) \in X_0, u(t) \in U \quad (7.1)$$

where $x(t) \in X$ is the position of the robot on the plane, $X \subseteq \mathbb{R}^2$ is the free workspace of the robot and $X_0 \subseteq X$ is a compact set that represents the set of initial positions. Note that $x : \mathbb{R}_{\geq 0} \rightarrow X$ is a continuous-time signal as introduced in Section 2.1.2. Here, we assume that initially the robot is at rest, i.e., $\dot{x}(0) = 0$, and that $U = \{u \in \mathbb{R}^2 \mid \|u\| \leq \mu\}$ where $\mu \in \mathbb{R}_{>0}$ models the constraints on the control input (forces or acceleration) and $\|\cdot\|$ is the Euclidean norm. The goal of this chapter is to construct a hybrid controller that generates control inputs $u(t)$ for system Σ so that for the set of initial states X_0 , the resulting motion $x(t)$ satisfies a formula-specification ϕ in the

propositional temporal logic over the reals, that is, LTL interpreted over continuous-time signals.

For the high level planning problem, we consider the existence of a number of regions of interest to the user. Such regions could be rooms and corridors in an indoor environment or areas to be surveyed in an outdoor environment. Let $AP = \{p_0, p_1, \dots, p_n\}$ be a finite set of symbols (atomic propositions) that label these areas. We reserve the symbol p_0 to model the free workspace of the robot, i.e., $\mathcal{O}(p_0) = X$.

In order to make apparent the use of LTL for the composition of motion planning specifications, we first present some examples. The propositional temporal logic over the reals can describe the usual properties of interest for control problems, i.e., *reachability* ($\diamond p$) and *safety*: ($\Box p$ or $\Box \neg p$). Beyond the usual properties, LTL can capture sequences of events and infinite behaviours:

- **Reachability while avoiding regions:** The formula $\neg(p_1 \vee p_2 \vee \dots \vee p_n) \mathcal{U} p_{n+1}$ expresses the property that the sets $\mathcal{O}(p_i)$ for $i = 1, \dots, n$ should be avoided until $\mathcal{O}(p_{n+1})$ is reached.
- **Sequencing:** The requirement that we must visit $\mathcal{O}(p_1)$, $\mathcal{O}(p_2)$ and $\mathcal{O}(p_3)$ in that order is captured by the formula $\diamond(p_1 \wedge \diamond(p_2 \wedge \diamond p_3))$.
- **Coverage:** Formula $\diamond p_1 \wedge \diamond p_2 \wedge \dots \wedge \diamond p_n$ reads as the system will eventually reach $\mathcal{O}(p_1)$ and eventually $\mathcal{O}(p_2)$ and ... eventually $\mathcal{O}(p_n)$, requiring the system to eventually visit all regions of interest without imposing any ordering.
- **Recurrence (Liveness):** The formula $\Box(\diamond p_1 \wedge \diamond p_2 \wedge \dots \wedge \diamond p_n)$ requires that the trajectory does whatever the coverage does and, in addition, will force the system to repeat the desired objective infinitely often.

More complicated specifications can be composed from the basic specifications using

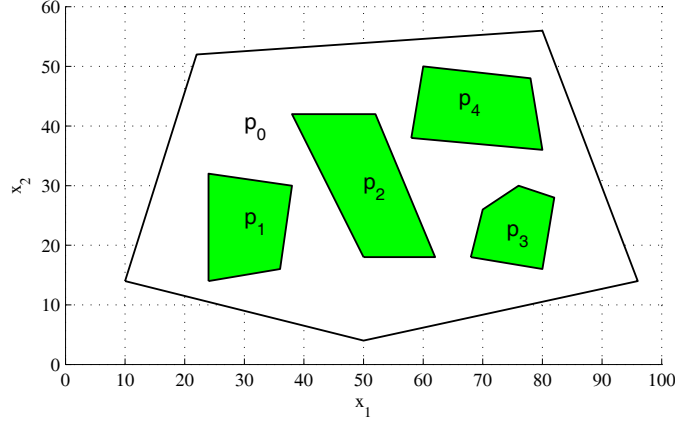


Figure 7.1: The simple environment of Example 7.1.1. The four regions of interest p_1, p_2, p_3, p_4 are enclosed by the polygonal region labeled by p_0 .

the logic operators. In order to better explain the different steps in our framework, we consider throughout the paper the following example.

Example 7.1.1. *Consider a robot that is moving in a convex polygonal environment p_0 with four areas of interest denoted by p_1, p_2, p_3, p_4 (see Fig. 7.1). Initially, the robot is placed somewhere in the region labeled by p_1 and its velocity is set to zero. The robot must accomplish the following task : “Visit area $\mathcal{O}(p_2)$, then area $\mathcal{O}(p_3)$, then area $\mathcal{O}(p_4)$ and, finally, return to and stay in region $\mathcal{O}(p_1)$ while avoiding areas $\mathcal{O}(p_2)$ and $\mathcal{O}(p_3)$ ”. This specification can be formally written as the LTL formula:*

$$\psi_1 = \Box p_0 \wedge \Diamond(p_2 \wedge \Diamond(p_3 \wedge \Diamond(p_4 \wedge (\neg(p_2 \vee p_3)) \mathcal{U} \Box p_1)))$$

Also, it is implied that the robot should always remain inside the free workspace X , i.e., region $\mathcal{O}(p_0)$, and that $X_0 = \mathcal{O}(p_1)$.

In this paper, for such specifications, we provide a computational solution to the following problem.

Problem 7.1.1. *Given the system Σ , an LTL formula ϕ and an observation map*

\mathcal{O} , construct a hybrid controller H_ϕ for Σ such that the trajectories of the closed-loop system $\bar{\Sigma}_\phi$ satisfy formula ϕ , i.e., $\mathcal{L}(\bar{\Sigma}_\phi) \subseteq \mathcal{L}(\phi, \mathcal{O})$.

We propose a hierarchical synthesis approach which consists of three components : tracking control using approximate simulation relations [79], robust satisfaction of LTL formulas [60, 58] and hybrid control for motion planning [62, 61]. First, Σ is abstracted to the first order system Σ' (*kinematics model*):

$$\dot{z}(t) = v(t), \quad z(t) \in Z, \quad z(0) \in Z_0, \quad v(t) \in V \quad (7.2)$$

where $z(t) \in Z$ is the position of the robot in the kinematics model, $Z \subseteq \mathbb{R}^2$ is a modified free workspace, $Z_0 = X_0$ is the set of possible initial positions and $V = \{v \in \mathbb{R}^2 \mid \|v\| \leq \nu\}$ for some $\nu \in \mathbb{R}_{>0}$ is the set of control input values (allowed velocity values). Using the notion of approximate simulation relation, we evaluate the precision δ with which the system Σ is able to track the trajectories of the abstraction Σ' and design a continuous tracking controller that we call *interface*. Secondly, from the LTL formula ϕ and the precision δ , we derive a more “robust” formula ϕ' such that if a trajectory z satisfies ϕ' , then any trajectory x remaining at time t within distance δ from $z(t)$ satisfies the formula ϕ . Thirdly, we design a hybrid controller $H_{\phi'}$ for the abstraction Σ' , so that the trajectories of the closed loop system satisfy the formula ϕ' . Finally, by putting these three components together, as shown in Fig. 7.2, we design a hybrid controller H_ϕ solving Problem 7.1.1. In the following, we detail each step of our approach.

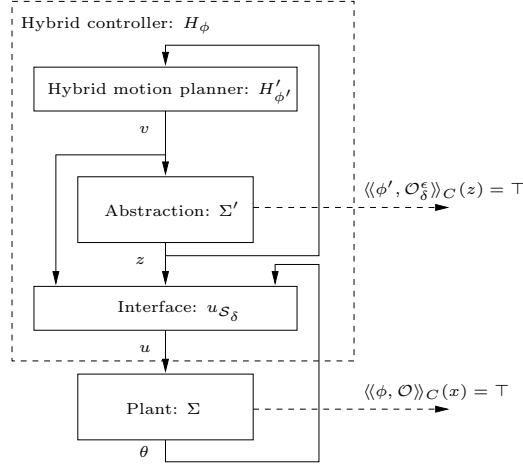


Figure 7.2: Hierarchical architecture of the hybrid controller H_ϕ .

7.2 Tracking using Approximate Simulation

In this section, we present a framework for tracking control with guaranteed error bounds. It allows us to design an interface between the dynamics model Σ and its kinematics abstraction Σ' so that Σ is able to track the trajectories of Σ' with a given precision. It is based on the notion of approximate simulation relation [82]. Whereas exact simulation relations require the observations, i.e., $x(t)$ and $z(t)$, of two systems to be identical, approximate simulation relations allow them to be different provided their distance remains bounded by some parameter.

Let us first rewrite the 2nd order model Σ as a system of 1st order differential equations

$$\Sigma : \begin{cases} \dot{x}(t) = y(t), & x(t) \in X, x(0) \in X_0 \\ \dot{y}(t) = u(t), & y(t) \in \mathbb{R}^2, y(0) = [0 \ 0]^T \end{cases}$$

where x is the position of the mobile robot and y its velocity. If we let $\theta = [x^T \ y^T]^T$, i.e., $\theta : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^4$, with $\theta(0) \in \Theta_0 = X_0 \times \{(0, 0)\}$, then

$$\dot{\theta} = A\theta + Bu \quad \text{and} \quad x = C_x\theta, \quad y = C_y\theta$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad C_y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, we review some of the definitions of Section 4.2 using the notation and the systems of the current chapter. First, we restate the definition of the approximate simulation relation.

Definition 7.2.1 (Simulation Relation). *A relation $\mathcal{S}_\delta \subseteq \mathbb{R}^2 \times \mathbb{R}^4$ is an approximate simulation relation of precision δ of Σ' by Σ if for all $(z_0, \theta_0) \in \mathcal{S}_\delta$,*

1. $\|z_0 - C_x \theta_0\| \leq \delta$
2. *For all state trajectories z of Σ' such that $z(0) = z_0$ there exists a state trajectory θ of Σ such that $\theta(0) = \theta_0$ and $\forall t \geq 0, (z(t), \theta(t)) \in \mathcal{S}_\delta$.*

An interface associated to the approximate simulation relation \mathcal{S}_δ allows to choose the input of Σ so that the states of Σ' and Σ remain in \mathcal{S}_δ .

Definition 7.2.2 (Interface). *A continuous function $u_{\mathcal{S}_\delta} : V \times \mathcal{S}_\delta \rightarrow U$ is an interface associated with an approximate simulation relation \mathcal{S}_δ , if for all $(z_0, \theta_0) \in \mathcal{S}_\delta$, for all trajectories z of Σ' associated with a given input signal v such that $z(0) = z_0$, the trajectory θ of Σ starting at $\theta(0) = \theta_0$ given by*

$$\dot{\theta}(t) = A\theta(t) + Bu_{\mathcal{S}_\delta}(v(t), z(t), \theta(t)) \tag{7.3}$$

satisfies for all $t \geq 0, (z(t), \theta(t)) \in \mathcal{S}_\delta$.

Thus, by interconnecting Σ and Σ' through the interface $u_{\mathcal{S}_\delta}$ as shown on Fig. 7.2, the system Σ tracks the trajectories of the abstraction Σ' with precision δ .

Proposition 7.2.1. *Let $\theta_0 \in \Theta_0$ and $z_0 = C_x \theta_0 \in Z_0$ such that $(z_0, \theta_0) \in \mathcal{S}_\delta$, then for all trajectories z of Σ' associated with a given input signal v and the initial state z_0 , the trajectory θ of Σ given by (7.3) for $\theta(0) = \theta_0$, satisfies for all $t \geq 0$, $\|C_x \theta(t) - z(t)\| \leq \delta$.*

Let us remark that the choice of the initial state z_0 of the abstraction Σ' is not independent of the initial state θ_0 of the system Σ ($z_0 = C_x \theta_0$).

Remark 7.2.1. *Usual hierarchical control approaches assume that the plant Σ is simulated by its abstraction Σ' . Here, the contrary is assumed. The abstraction Σ' is (approximately) simulated by the plant Σ : the approximate simulation relation is used as a tool for tracking controller design.*

The construction of approximate simulation relations can be done effectively using a class of functions called simulation functions [82]. Essentially, a simulation function of Σ' by Σ is a positive function bounding the distance between the observations and non-increasing under the parallel evolution of the systems.

Definition 7.2.3 (Simulation Function). *Let $\mathcal{F} : \mathbb{R}^2 \times \mathbb{R}^4 \rightarrow \mathbb{R}_{\geq 0}$ be a continuous and piecewise differentiable function. Let $u_{\mathcal{F}} : V \times \mathbb{R}^2 \times \mathbb{R}^4 \rightarrow \mathbb{R}^2$ be a continuous function. \mathcal{F} is a simulation function of Σ' by Σ , and $u_{\mathcal{F}}$ is an associated interface if for all $(z, \theta) \in \mathbb{R}^2 \times \mathbb{R}^4$, the following two inequalities hold*

$$\mathcal{F}(z, \theta) \geq \|z - C_x \theta\|^2 \quad (7.4)$$

$$\sup_{v \in V} \left(\frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) \right) \leq 0 \quad (7.5)$$

Then, approximate simulation relations can be defined as level sets of the simulation function.

Theorem 7.2.1. *Let the relation $\mathcal{S}_\delta \subseteq \mathbb{R}^2 \times \mathbb{R}^4$ be given by*

$$\mathcal{S}_\delta = \{(z, \theta) \mid \mathcal{F}(z, \theta) \leq \delta^2\}.$$

If for all $v \in V$, for all $(z, \theta) \in \mathcal{S}_\delta$, we have $u_{\mathcal{F}}(v, z, \theta) \in U$, then \mathcal{S}_δ is an approximate simulation relation of precision δ of Σ' by Σ and $u_{\mathcal{S}_\delta} : V \times \mathcal{S}_\delta \rightarrow U$ given by $u_{\mathcal{S}_\delta}(v, z, \theta) = u_{\mathcal{F}}(v, z, \theta)$ is an associated interface.

Now we are in position to state the result that will enable us to perform tracking control.

Proposition 7.2.2. *Assume that for the systems Σ and Σ' the constraints μ and ν satisfy the inequality*

$$\frac{\nu}{2} (1 + |1 - 1/\alpha| + 2/\sqrt{\alpha}) \leq \mu \tag{7.6}$$

for some $\alpha > 0$. Then,

$$\mathcal{S}_\delta = \{(z, \theta) \mid \mathcal{F}(z, \theta) \leq 4\nu^2\}$$

where $\mathcal{F}(z, \theta) = \max(Q(z, \theta), 4\nu^2)$ with

$$Q(z, \theta) = \|C_x\theta - z\|^2 + \alpha\|C_x\theta - z + 2C_y\theta\|^2$$

is an approximate simulation relation of precision $\delta = 2\nu$ of Σ' by Σ and an associated interface is

$$u_{\mathcal{S}_\delta}(v, z, \theta) = \frac{v}{2} + \frac{-1 - \alpha}{4\alpha}(C_x\theta - z) - C_y\theta.$$

The importance of Proposition 7.2.2 is the following. Assume that the initial state of the abstraction Σ' is chosen so that $z(0) = C_x\theta(0)$ and that Σ' and Σ are interconnected through the interface $u_{\mathcal{S}_\delta}$. Then, from Theorem 7.2.1, the observed trajectories $x(t)$ of system Σ track the trajectories $z(t)$ of Σ' with precision 2ν .

The parameter α in the simulation function in Proposition 7.2.2 can be thought as a Lagrange multiplier. Basically, if α is large, then the term $\|C_x\theta - z + 2C_y\theta\|$ gets penalized and, as a consequence, this term will be small. On the other hand, if α is small, then the term $\|C_x\theta - z\|$ is penalized and, thus, this term will become small. The parameter α has also the following influence on the interface. If α is large, then the dynamics are smooth since $\lim_{\alpha \rightarrow \infty} (1 + \alpha)/4\alpha = 1/4$, while when α is small the dynamics are stiff since $\lim_{\alpha \rightarrow 0^+} (1 + \alpha)/4\alpha = +\infty$.

Remark 7.2.2. *In this section we have only imposed bounds on \ddot{x} through the constraint μ . Bounds on the actual value of x are imposed through the allowed free workspace X . If bounds on the velocity \dot{x} of Σ are also required, then these can be computed from the equation of the interface $u_{\mathcal{S}_\delta}$ since v , \ddot{x} and $\|x(t) - z(t)\|$ are bounded. If the bounds on \dot{x} are not satisfied, then we can adjust μ and ν accordingly.*

7.3 Robust Interpretation of LTL Formulas

In the previous section, we designed a control interface which enables the dynamic model Σ to track its abstract kinematic model Σ' with accuracy $\delta = 2\nu$. In this brief section, we demonstrate how we can utilize the observation map \mathcal{O}_δ^e (see Section 3.1.2) in order take into account the bound δ on the tracking error.

Example 7.3.1. *Revisiting Example 7.1.1, we need first to modify the input specifi-*

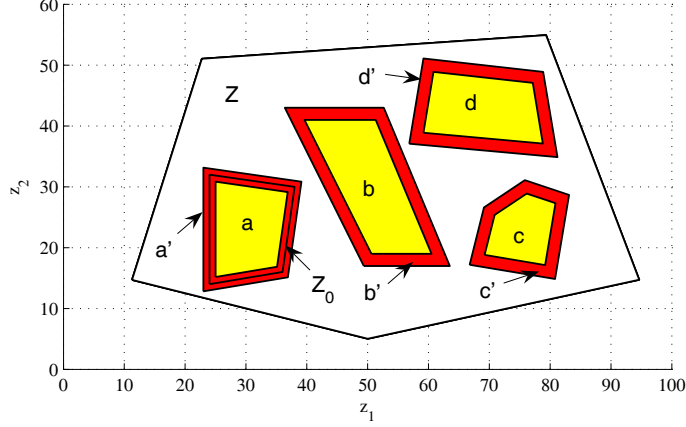


Figure 7.3: The modified workspace of Example 7.3.1 for $\delta = 1 + \varepsilon$ with ε sufficiently small.

ation ψ_1 . The formula $\mathbf{nnf}(\psi_1)$ is equal to

$$\mathbf{nnf}(\psi_1) = \Box p_0 \wedge \Diamond(p_2 \wedge \Diamond(p_3 \wedge \Diamond(p_4 \wedge (\neg p_2 \wedge \neg p_3) \mathcal{U} \Box p_1)))$$

and the corresponding formula $\psi'_1 = \mathbf{pos}(\mathbf{nnf}(\psi_1))$ is

$$\psi'_1 = \Box \xi_{p_0} \wedge \Diamond(\xi_{p_2} \wedge \Diamond(\xi_{p_3} \wedge \Diamond(\xi_{p_4} \wedge (\xi_{\neg p_2} \wedge \xi_{\neg p_3}) \mathcal{U} \Box \xi_{p_1}))).$$

Note that after the translation no negation operator appears in the formula. We can now apply the contraction operation on the regions of interest labeled by Ξ_{AP} and the free workspace X and derive the δ -robust interpretation of the propositions in Ξ_{AP} and the modified workspace Z (see Fig. 7.3). For the purposes of this example, we define the map $h_\delta : Z \rightarrow \mathcal{P}(\Xi_{AP})$ such that for any $z \in Z$ we have $h_\delta(z) = \{\xi \in \Xi_{AP} \mid z \in \mathcal{O}_\delta^e(\xi)\}$. Any point z in region a (yellow or light gray) is labeled by the set of propositions $h_\delta(z) = \{\xi_{p_0}, \xi_{p_1}, \xi_{\neg p_2}, \xi_{\neg p_3}, \xi_{\neg p_4}\}$, while any point z in the annulus region a' (red or dark gray) is labeled by the set $h_\delta(z) = \{\xi_{p_0}, \xi_{\neg p_2}, \xi_{\neg p_3}, \xi_{\neg p_4}\}$. Notice that $Z = \mathcal{O}_\delta^e(\xi_{p_0})$ and that $Z_0 = X_0 = \mathcal{O}_\delta^e(p_1)$.

Remark 7.3.1. *The δ -contraction of a polyhedral set is not always a polyhedral set. In order to maintain a polyhedral description for all the sets, we under-approximate the δ -contraction by the inward δ -offset. Informally, the inward δ -offset of a polyhedral set is the inward δ -displacement of its facets along the corresponding normal directions. Since the δ -offset is an under-approximation of the δ -contraction, Corollary 7.3.1 still holds.*

The following corollary of Theorems 3.1.2 and 3.1.1 is the connecting link between the specifications satisfied by the abstraction Σ' and the concrete system Σ . Informally, it states that given $\delta > 0$ if a trajectory z of Σ' satisfies the δ -robust interpretation of the input specification ϕ and the trajectories z and x always remain δ -close, then x will satisfy the same non-robust specification ϕ .

Corollary 7.3.1. *Consider a formula $\phi \in LTL_{\mathbb{B}}(AP)$, a map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a number $\delta \in \mathbb{R}_{>0}$, then for all trajectories x of Σ and z of Σ' such that $\|z(t) - x(t)\| < \delta$ for all $t \geq 0$, we have $\langle\langle \mathbf{pos}(\mathbf{nf}(\phi)), \mathcal{O}_{\delta}^e \rangle\rangle_C(z) = \top$ implies $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(x) = \top$.*

It is easy to see that $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(x) = \top$ does not imply $\langle\langle \mathbf{pos}(\mathbf{nf}(\phi)), \mathcal{O}_{\delta}^e \rangle\rangle_C(z)$. For the sake of example, consider the simple specification $\phi = \diamond p$ and a trajectory x that enters $\mathcal{O}(p)$ less than δ deep.

7.4 Temporal Logic Motion Planning

Having presented the connection between the dynamics model Σ and the kinematics model Σ' , we proceed to solve the temporal logic motion planning problem for Σ' . Formally, we solve the following more general problem.

Problem 7.4.1. *Given the system Σ' , a set of atomic propositions Ξ_{AP} , an LTL formula $\phi' \in LTL_{\mathbb{B}}^+(\Xi_{AP})$ and a map $\mathcal{O}_{\delta}^e : \Xi_{AP} \rightarrow \mathcal{P}(Z)$, construct a hybrid con-*

troller $H'_{\phi'}$ such that for all trajectories $z(t)$ of Σ' under controller $H'_{\phi'}$ we have $\langle\langle\phi', \mathcal{O}_\delta^e\rangle\rangle_C(z) = \top$.

Our solution consists of the following three steps: (1) reduction of the continuous environment Z into a discrete graph, (2) temporal logic planning over the discrete graph, and (3) continuous implementation of the final discrete plan.

7.4.1 Discrete Abstraction of Robot Motion

In order to use discrete logics to reason about continuous systems, we need to construct a finite partition of the continuous state space Z with respect to the map \mathcal{O}_δ^e . For that purpose, we can use many efficient cell decomposition methods for polygonal environments [35, 125]. The choice of a particular decomposition method depends on the type of controllers that we wish to use. The affine controllers presented in [21] require a triangular decomposition [53], while the potential field controllers of [43] can be applied to any convex cell that can be represented by a polygon [110]. Note that by employing any workspace decomposition method we can actually construct a *topological graph* (or *roadmap* as it is sometimes referred to [125]). A topological graph describes which cells are topologically adjacent, i.e., each node in the graph represents a cell and each edge in the graph implies topological adjacency of the cells. No matter what decomposition algorithm is employed, the workspace's decomposition must be *proposition preserving* with respect to the mapping \mathcal{O}_δ^e . In other words, we require that all the points which belong in the same cell must be labeled by the same set of propositions.

On a more formal note, we can partition the workspace Z with respect to the map \mathcal{O}_δ^e and the set of initial conditions Z_0 into a number of equivalence classes, that is, into a number of sets of points which satisfy the same property (in this case the

same set of propositions). Note that mathematically the set of equivalence classes consists of the interior of the cells, the edges and the vertices. In the following, we define $\mathcal{Q} = \{q_1, \dots, q_n\}$ to be the set of all equivalence classes. Let us introduce the map $T : Z \rightarrow \mathcal{Q}$ which sends each state $z \in Z$ to one equivalence class in \mathcal{Q} . Then, we can formally state the proposition preserving property as follows : $\forall z_i, z_j \in Z . T(z_i) = T(z_j)$ implies $h_\delta(z_i) = h_\delta(z_j)$. Moreover, we define the “inverse” map $T^{-1} : \mathcal{Q} \rightarrow \mathcal{P}(Z)$ of T such that $T^{-1}(q)$ maps to all states $z \in Z$ which are contained in the equivalence class q .

In the following paragraphs, we present how the topological graph resulting from the decomposition of Z with respect to \mathcal{O}_δ^e can be converted into a Finite Transition System (FTS) that serves as an abstract model of the robot motion. Posing the topological graph as an FTS allows us to use standard automata theoretic techniques [41] in order to solve the high level planning problem.

Definition 7.4.1 (FTS). *The Finite Transition System that abstracts the continuous dynamics is the tuple $\mathcal{D} = (Q, Q_0, \rightarrow_{\mathcal{D}}, h_{\mathcal{D}}, \Xi_{AP})$ where:*

- Q is a set of states. Here, $Q \subseteq \mathcal{Q}$ is the set of equivalence classes that represent the interior of the cells.
- $Q_0 \subseteq Q$ is the set of possible initial cells. Here, Q_0 satisfies $\cup_{q_0 \in Q_0} \overline{T^{-1}(q_0)} = Z_0$.
- $\rightarrow_{\mathcal{D}} \subseteq Q \times Q$ captures the topological relationship between the cells. There is a transition from cell q_i to cell q_j written as $q_i \rightarrow_{\mathcal{D}} q_j$ if the cells labelled by q_i, q_j are adjacent, i.e., $\overline{T^{-1}(q_i)}$ and $\overline{T^{-1}(q_j)}$ share a common edge. Finally, for all $q \in Q$ we add a self-loop, i.e., $q \rightarrow_{\mathcal{D}} q$.
- $h_{\mathcal{D}} : Q \rightarrow \mathcal{P}(\Xi_{AP})$ is a map defined as $h_{\mathcal{D}}(q) = \{\xi \in \Xi_{AP} \mid T^{-1}(q) \subseteq \mathcal{O}_\delta^e(\xi)\}$.

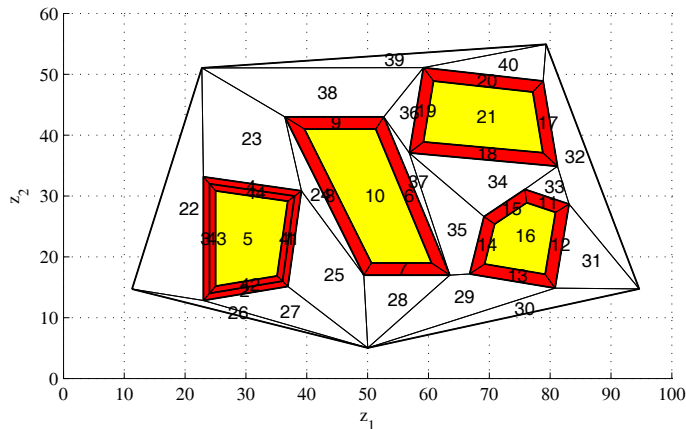


Figure 7.4: Convex cell decomposition (Example 7.4.1).

We define a *path* on the FTS to be a sequence of states (cells) and a *trace* to be the corresponding sequence of sets of propositions. Formally, a path is a discrete-time signal $\gamma : \mathbb{N} \rightarrow Q$ such that for each $i \in \mathbb{N}$ we have $\gamma(i) \rightarrow_{\mathcal{D}} \gamma(i+1)$ and the corresponding trace is the function composition $\tilde{\gamma} = h_{\mathcal{D}} \circ \gamma : \mathbb{N} \rightarrow \mathcal{P}(\Xi_{AP})$.

Example 7.4.1. *To better explain the notion of a proposition preserving cell decomposition, let us consider the convex decomposition of the workspace of Example 7.3.1 which appears in Fig. 7.4. The topological graph contains 40 nodes and 73 edges. Notice that all the points in the interior of each cell satisfy the same set of propositions. For the following examples, we let \mathcal{D}_1 denote the FTS which corresponds to the aforementioned topological graph.*

7.4.2 Linear Temporal Logic Planning

The transition system \mathcal{D} , which was constructed in the previous section, will serve as an abstract model of the robot's motion. In this work, we are interested in the construction of automata that only accept the traces of \mathcal{D} which satisfy the LTL formula ϕ' . Such automata (which are referred to as Büchi automata [41, §9.1]) differ from the classic finite automata [41, §9.1] in that they accept infinite strings (traces

of \mathcal{D} in our case).

Definition 7.4.2 (Büchi Automaton). *A Büchi automaton is a tuple $\mathcal{B} = (S_{\mathcal{B}}, s_{0\mathcal{B}}, \Omega, \lambda_{\mathcal{B}}, F_{\mathcal{B}})$ where:*

- $S_{\mathcal{B}}$ is a finite set of states and $s_{0\mathcal{B}}$ is the initial state.
- Ω is an input alphabet.
- $\lambda_{\mathcal{B}} : S_{\mathcal{B}} \times \Omega \rightarrow \mathcal{P}(S_{\mathcal{B}})$ is a transition relation.
- $F_{\mathcal{B}} \subseteq S_{\mathcal{B}}$ is a set of accepting states.

In order to define what it means for a Büchi automaton to accept a trace, we must first introduce some terminology. A *run* r of \mathcal{B} is the sequence of states $r : \mathbb{N} \rightarrow S_{\mathcal{B}}$ that occurs under an input trace $\tilde{\gamma}$, that is for $i = 0$ we have $r(0) = s_{0\mathcal{B}}$ and for all $i \geq 0$ we have $r(i+1) \in \lambda_{\mathcal{B}}(r(i), \tilde{\gamma}(i))$. Let $\text{lim}(\cdot)$ be the function that returns the set of states that are encountered infinitely often in the run r of \mathcal{B} . Then, a run r of a Büchi automaton \mathcal{B} over an infinite trace $\tilde{\gamma}$ is *accepting* if and only if $\text{lim}(r) \cap F_{\mathcal{B}} \neq \emptyset$. Informally, a run r is accepting when some accepting state $s \in F_{\mathcal{B}}$ appears in r infinitely often. Finally, we define the language $\mathcal{L}(\mathcal{B})$ of \mathcal{B} to be the set of all traces $\tilde{\gamma}$ that have a run that is accepted by \mathcal{B} .

For each LTL formula ϕ' , we can construct a Büchi automaton

$$\mathcal{B}_{\phi'} = (S_{\mathcal{B}_{\phi'}}, s_{0\mathcal{B}_{\phi'}}, \mathcal{P}(\Xi_{AP}), \lambda_{\mathcal{B}_{\phi'}}, F_{\mathcal{B}_{\phi'}})$$

that accepts the infinite traces which satisfy the specification ϕ' , i.e., $\tilde{\gamma} \in \mathcal{L}(\mathcal{B}_{\phi'})$ iff $\langle\langle \phi', \mathcal{O}_D \rangle\rangle_D(\gamma) = \top$. Here, the observation map $\mathcal{O}_D : \Xi_{AP} \rightarrow \mathcal{P}(Q)$ is defined as

$$\forall \xi \in \Xi_{AP} . \mathcal{O}_D(\xi) = \{q \in Q \mid T^{-1}(q) \subseteq \mathcal{O}_\delta^e(\xi)\}.$$

The translation from an LTL formula ϕ' to a Büchi automaton \mathcal{B}'_ϕ is a well studied problem and, thus, we refer the reader to [41, §9.4] and the references therein for the theoretical details behind this translation.

We can now use the abstract representation of robot's motion, that is the FTS, in order to reason about the desired motion of the robot. First, we convert the FTS \mathcal{D} into a Büchi automaton \mathcal{D}' . The translation from \mathcal{D} to \mathcal{D}' enables us to use standard tools and techniques from automata theory [41, §9] alleviating, thus, the need for developing new theories. Translating an FTS into an automaton is a standard procedure which can be found in any formal verification textbook (see [41, §9.2]). The procedure consists of (i) adding a dummy initial state $q_d \notin Q$ that has one transition to each state q_0 in Q_0 , (ii) moving the label from each state q to all of its incoming transitions, and (iii) making all the states accepting.

Definition 7.4.3 (FTS to Automaton). *The Büchi automaton \mathcal{D}' which corresponds to the FTS \mathcal{D} is the automaton $\mathcal{D}' = (Q', q_d, \mathcal{P}(\Xi_{AP}), \lambda_{\mathcal{D}'}, F_{\mathcal{D}'})$ where:*

- $Q' = Q \cup \{q_d\}$ for $q_d \notin Q$.
- $\lambda_{\mathcal{D}'} : Q' \times \mathcal{P}(\Xi_{AP}) \rightarrow \mathcal{P}(Q')$ is the transition relation defined as: $q_j \in \lambda_{\mathcal{D}'}(q_i, l)$ iff $q_i \rightarrow_{\mathcal{D}} q_j$ and $l = h_{\mathcal{D}}(q_j)$ and $q_0 \in \lambda_{\mathcal{D}'}(q_d, l)$ iff $q_0 \in Q_0$ and $l = h_{\mathcal{D}}(q_0)$.
- $F_{\mathcal{D}'} = Q'$ is the set of accepting states.

Similarly to \mathcal{B} , we define the language $\mathcal{L}(\mathcal{D}')$ of \mathcal{D}' to be the set of all possible traces that are accepted by \mathcal{D}' . Note that any path generated by \mathcal{D} has a trace that belongs to $\mathcal{L}(\mathcal{D}')$.

Now that all the related terminology is defined, we can give an overview of the basic steps involved in the temporal logic planning [74]. Our goal in this section is to generate paths on \mathcal{D} that satisfy the specification ϕ' . In automata theoretic terms,

we want to find the subset of the language $\mathcal{L}(\mathcal{D}')$ which also belongs to the language $\mathcal{L}(B_{\phi'})$. This subset is simply the intersection of the two languages $\mathcal{L}(\mathcal{D}') \cap \mathcal{L}(B_{\phi'})$ and it can be constructed by taking the product $\mathcal{D}' \times B_{\phi'}$ of the Büchi automaton \mathcal{D}' and the Büchi automaton $B_{\phi'}$. Informally, the Büchi automaton $B_{\phi'}$ restricts the behavior of the system \mathcal{D}' by permitting only certain acceptable transitions. Then, given an initial state in the FTS \mathcal{D} , which is an abstraction of the actual initial position of the robot, we can choose a particular trace from $\mathcal{L}(\mathcal{D}) \cap \mathcal{L}(B_{\phi'})$ according to a preferred criterion. In the following, we present the details of this construction.

Definition 7.4.4 (Product). *The product automaton $\mathcal{A} = \mathcal{D}' \times B_{\phi'}$ is the automaton $\mathcal{A} = (S_{\mathcal{A}}, s_{0\mathcal{A}}, \mathcal{P}(\Xi_{AP}), \lambda_{\mathcal{A}}, F_{\mathcal{A}})$ where:*

- $S_{\mathcal{A}} = Q' \times S_{B_{\phi'}}$ and $s_{0\mathcal{A}} = \{(q_d, s_{0B_{\phi'}})\}$.
- $\lambda_{\mathcal{A}} : S_{\mathcal{A}} \times \mathcal{P}(\Xi_{AP}) \rightarrow \mathcal{P}(S_{\mathcal{A}})$ such that $(q_j, s_j) \in \lambda_{\mathcal{A}}((q_i, s_i), l)$ iff $q_j \in \lambda_{\mathcal{D}'}(q_i, l)$ and $s_j \in \lambda_{B_{\phi'}}(s_i, l)$.
- $F_{\mathcal{A}} = Q' \times F$ is the set of accepting states.

By construction, the following theorem is satisfied (recall that $\tilde{\gamma}$ is a trace of \mathcal{D} if and only if $\tilde{\gamma}$ is accepted by \mathcal{D}').

Lemma 7.4.1 (Adapted from [74]). *A trace $\tilde{\gamma}$ of \mathcal{D} that satisfies the specification ϕ' exists iff the language of \mathcal{A} is non-empty, i.e., $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D}') \cap \mathcal{L}(B_{\phi'}) \neq \emptyset$.*

Checking the emptiness of language $\mathcal{L}(\mathcal{A})$ is an easy algorithmic problem [41, §9.3]. First, we convert automaton \mathcal{A} to a directed graph and, then, we find the strongly connected components (SCC) [44, §22.5] in that graph. If at least one SCC that contains an accepting state is reachable from $s_{0\mathcal{A}}$, then the language $\mathcal{L}(\mathcal{A})$ is not empty. The rationale behind this construction is that any infinite path on a

finite graph must visit at least one node of the graph infinitely often. However, we are not just interested in figuring out whether $\mathcal{L}(\mathcal{A}) = \emptyset$. We need to construct an accepting run of \mathcal{A} and from that to derive a discretized path for the robot on \mathcal{D} . The good news are that if $\mathcal{L}(\mathcal{A})$ is nonempty, then there exist accepting (infinite) runs on \mathcal{A} that have a finite representation. Each such run consists of two parts. The first part is a *finite* sequence of states $r(0)r(1)\dots r(m_f)$ which corresponds to the sequence of states starting from $r(0) = s_{0\mathcal{A}}$ and reaching a state $r(m_f) \in F_{\mathcal{A}}$. The second part is a *periodic* sequence of states $r(m_f)r(m_f + 1)\dots r(m_f + m_l)$ such that $r(m_f + m_l) = r(m_f)$ which corresponds to the part of the run that traverses some part of the strongly connected component. Here, $m_f, m_l \geq 0$ is less than or equal to the number of states in \mathcal{D} , i.e., $m_f, m_l \leq |Q|$.

Since in this paper we are concerned with a path planning application, it is desirable to choose an accepting run that traverses as few different states on \mathcal{D} as possible. For example, we could select an accepting run with a finite representation of a minimal size. The heuristic rule that we employ for the construction of such a run is to find the accepting run with the shortest finite part and the shortest periodic part. The high level description of the algorithm is as follows. First, we find all the shortest sequences of states from $s_{0\mathcal{A}}$ to all the accepting states in $F_{\mathcal{A}}$ using Breadth First Search (BFS) [44, §22.2]. The running time of BFS is linear in the number of states and the number of transitions in \mathcal{A} . Then, from each reachable accepting state $q_a \in F_{\mathcal{A}}$ we initiate a new BFS in order to find the shortest sequence of states that leads back to q_a . Note that if no accepting state is reachable from $s_{0\mathcal{A}}$ or no infinite loop can be found, then the language $\mathcal{L}(\mathcal{A})$ is empty and, hence, the temporal logic planning problem does not have a solution. Moreover, if $\mathcal{L}(\mathcal{A}) \neq \emptyset$, then this algorithm can potentially return a set R of accepting runs r each leading to a different accepting state in $F_{\mathcal{A}}$ with a different periodic part. From the set of runs R , we can

easily derive a corresponding set of paths Γ on \mathcal{D} such that for all $\gamma \in \Gamma$ we have that the trace $\tilde{\gamma}$ satisfies ϕ' .

Proposition 7.4.1. *Let $pr : S_{\mathcal{A}} \rightarrow Q$ be the projection on Q , i.e., $pr(q, s) = q$. If r is an accepting run of \mathcal{A} , then $\gamma = (pr \circ r)|_1$ is a path on \mathcal{D} such that $\langle\langle \phi', \mathcal{O}_{\mathcal{D}} \rangle\rangle_{\mathcal{D}}(\gamma) = \top$.*

Any path $\gamma \in \Gamma$ can be characterized by a pair of sequences of states (γ^f, γ^l) . Here, $\gamma^f = \gamma_1^f \gamma_2^f \dots \gamma_{n_f}^f$ denotes the non-periodic part of the path and $\gamma^l = \gamma_1^l \gamma_2^l \dots \gamma_{n_l}^l$ the periodic part (infinite loop) such that $\gamma_{n_f}^f = \gamma_1^l$. The relation between the pair (γ^f, γ^l) and the path γ is given by $\gamma(i) = \gamma_{i+1}^f$ for $0 \leq i \leq n_f - 2$ and $\gamma(i) = \gamma_j^l$ with $j = ((i - n_f + 1) \bmod n_l) + 1$ for $i \geq n_f - 1$.

Example 7.4.2. *The Büchi automaton $\mathcal{B}_{\psi'_1}$ that accepts the paths that satisfy $\psi'_1 = \mathbf{pos}(\mathbf{nnf}(\psi_1))$ has 5 states (one accepting) and 13 transitions. For the conversion from LTL to Büchi automata, we use the python toolbox LTL2NBA by Fritz and Teegen, which is based on [70]. The product automaton $\mathcal{A}_1 = \mathcal{D}'_1 \times \mathcal{B}_{\psi'_1}$ has 205 states. The shortest path on the topological graph starting from cell 5 is: $(\gamma^f, \gamma^l) = (\{5, 41, 1, 25, 24, 8, 10, 6, 37, 35, 14, 16, 15, 34, 18, 21, 19, 36, 38, 23, 4, 44, 5\}, \{5\})$. Using Fig. 7.4, the reader can verify that this sequence satisfies ψ'_1 under the map \mathcal{O}_s^e . A prototype MATLAB implementation of the planning part of our framework took 0.61 sec for this example.*

The set Γ as computed above might not contain a path for every $q_0 \in Q_0$ and in certain cases some of the paths that were computed might not have a minimal finite representation. To see this, consider the runs which start from $s_{0,\mathcal{A}}$ and pass through the set of states $Q_0 \times S_{\mathcal{B}_{\phi'}}$. It is possible that all these runs converge to the same shortest sequence of states (γ^f) before reaching an accepting state. Since BFS constructs a tree, this implies that only one run would be the shortest and the rest

would either reach an accepting state at a longer distance or not reach an accepting state at all. One practical solution to this problem is to start a new BFS from each accepting state that belongs to an accepting cycle, i.e., the state q_a in the above high-level algorithm, and find which states in $Q_0 \times S_{\mathcal{B}_{\phi'}}$ are backward reachable from it. Then, we can repeat the procedure by starting from a different $q_a \in F_{\mathcal{A}}$ until all the states in Q_0 have been covered.

Remark 7.4.1. *Under the assumption that the initial workspace X is a connected space and due to the fact that the system Σ' models a fully actuated kinematic model of a robot, there can exist only two cases for which our planning method can fail to return a solution. First, when the workspace Z becomes disconnected or the sets $\mathcal{O}_\delta^e(\xi)$ for $\xi \in \Xi_{AP}$ become empty due to the dynamics of the system Σ , and second, when there exist logical inconsistencies in the temporal logic formula ϕ' with respect to the environment Z . As an example, consider a simple environment with two rooms connected through a corridor. The first case occurs when the corridor is contracted so much that no longer connects the two rooms. The second case occurs when the specification is : “Go from one room to the other while avoiding the corridor”.*

7.4.3 Continuous Implementation of Discrete Trajectory

Our next task is to utilize each discrete path $\gamma \in \Gamma$ in order to construct a hybrid control input $v(t)$ for $t \geq 0$ which will drive Σ' so that its trajectories $z(t)$ satisfy the LTL formula ϕ' . We achieve this desired goal by simulating (or implementing) at the continuous level each discrete transition of γ . This means that if the discrete system \mathcal{D} makes a transition $q_i \rightarrow_{\mathcal{D}} q_j$, then the continuous system Σ' must match this discrete step by moving from any position in the cell $\overline{T^{-1}(q_i)}$ to a position in the cell $\overline{T^{-1}(q_j)}$. Moreover, if the periodic part in the path γ consists of just a single

state q_t , then we have to guarantee that the position of the robot always remains in the invariant set $T^{-1}(q_t)$.

These basic control specifications imply that we need at least two types of continuous feedback control laws. We refer to these control laws as *reachability* and *cell invariant* controllers. Informally, a reachability controller drives each state inside a cell q to a predefined region on the cell's boundary, while the cell invariant controller guarantees that all the trajectories that start inside a cell q will always remain in that cell.

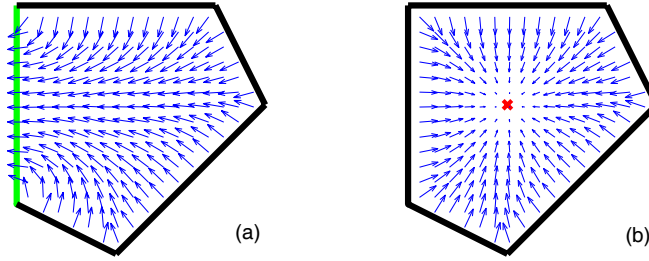


Figure 7.5: (a) Reachability and (b) Cell invariant controller.

Let us assume that we are given or that we can construct a finite collection of continuous feedback control laws $\{g_\kappa\}_{\kappa \in K}$ indexed by a control alphabet K such that for any $\kappa \in K$ we have $g_\kappa : Z_\kappa \rightarrow V$ with $Z_\kappa \subseteq Z$. In our setting, we make the following additional assumptions. First, we define the operational range of each controller to be one of the cells in the workspace of the robot, i.e., for any $\kappa \in K$ there exists for some $q \in Q$ such that $Z_\kappa = \overline{T^{-1}(q)}$. Second, if g_κ is a reachability controller, then we require that all the trajectories which start in Z_κ must converge on the same subset of the boundary of Z_κ within finite time while never exiting Z_κ before that time. Finally, if g_κ is a cell invariant controller, then we require that all the trajectories which initiate from a point in Z_κ converge on the barycenter b_κ of Z_κ . Examples of such feedback control laws for Σ' appear in Fig. 7.5. A formal presentation of these types of controllers is beyond the scope of this chapter and the

interested reader can find further details in [21, 43, 127].

The way we can compose such controllers given the pair (γ^f, γ^l) , which characterizes a path $\gamma \in \Gamma$, is as follows. First note that it is possible to get a finite repetition of states in the path γ , for example there can exist some $i \geq 0$ such that $\gamma(i) = \gamma(i+1)$ but $\gamma(i+1) \neq \gamma(i+2)$. This situation might occur because we have introduced self-loops in the automaton \mathcal{D}' in conjunction with possibility that the Büchi automaton $B_{\phi'}$ might not be optimal (in the sense of number of states and transitions). Therefore, we first remove finite repetitions¹ of states from γ . Next, we define the control alphabet to be $K = K^f \cup K^l \subseteq Q \times Q$ where $K^f = \cup_{i=1}^{n_f-1} \{(\gamma_i^f, \gamma_{i+1}^f)\} \cup \{(\gamma_{n_f}^f, \gamma_1^l)\}$ and $K^l = \cup_{i=1}^{n_l-1} \{(\gamma_i^l, \gamma_{i+1}^l)\} \cup \{(\gamma_{n_l}^l, \gamma_1^l)\}$ when $n_l > 1$ or $K^l = \emptyset$ otherwise. For any $\kappa = (q_i, q_j) \in K \setminus \{(\gamma_{n_f}^f, \gamma_1^l)\}$, we design g_κ to be a reachability controller that drives all initial states in $Z_\kappa = \overline{T^{-1}(q_i)}$ to the common edge $\overline{T^{-1}(q_i)} \cap \overline{T^{-1}(q_j)}$. Finally for $\kappa = (\gamma_{n_f}^f, \gamma_1^l)$, we let g_κ be a cell invariant controller for the cell γ_1^l .

It is easy to see now how we can use each pair (γ^f, γ^l) in order to construct a hybrid controller $H'_{\phi'}$. Starting anywhere in the cell $\overline{T^{-1}(\gamma_1^f)}$, we apply the control law $g_{(\gamma_1^f, \gamma_2^f)}$ until the robot crosses the edge $\overline{T^{-1}(\gamma_1^f)} \cap \overline{T^{-1}(\gamma_2^f)}$. At that point, we switch the control law to $g_{(\gamma_2^f, \gamma_3^f)}$. The above procedure is repeated until the last cell of the finite path γ^f at which point we apply the cell invariant controller $g_{(\gamma_{n_f}^f, \gamma_1^l)}$. If the periodic part γ^l of the path has only one state, i.e., $n_l = 1$, then this completes the construction of the hybrid controller $H'_{\phi'}$. If on the other hand $n_l > 1$, then we check whether the trajectory $z(t)$ has entered an ε -neighborhood of the barycenter of the cell invariant controller. If so, we apply ad infinitum the sequential composition of the controllers that correspond to the periodic part of the path γ^l followed by the cell invariant controller $g_{(\gamma_{n_f}^f, \gamma_1^l)}$.

¹Removing such repeated states from γ does not change the fact that $\langle\langle \phi', \mathcal{O}_D \rangle\rangle_D(\gamma) = \top$. This is true because LTL formulas without the *next* time operator are *stutter invariant* [41, §10].

The cell invariant controller is necessary in order to avoid Zeno behavior [131]. Since there can only exist at most one Zeno cycle in the final hybrid automaton and this cycle is guaranteed to not generate Zeno behaviors due to the existence of the cell invariant controller, the following proposition is immediate.

Proposition 7.4.2. *The trajectories z of the system $[\Sigma', H'_{\phi'}]$ satisfy the finite variability property.*

In the following, we denote the hybrid controller which corresponds to the path γ that satisfies formula ϕ' starting at position i on the path by $H'_{\phi'}(\gamma, i)$. If we use the whole path γ for the construction of the controller, then we just write $H'_{\phi'}(\gamma)$ for the corresponding hybrid controller. Assuming now that Σ' is controlled by the hybrid controller $H'_{\phi'}(\gamma)$ which is constructed as described above, we can prove the following theorem.

Theorem 7.4.1. *Let $\phi' \in LTL_{\mathbb{B}}^+(\Xi_{AP})$, Γ be a set of paths on \mathcal{D} such that $\forall \gamma \in \Gamma$ we have $\langle\langle \phi', \mathcal{O}_D \rangle\rangle_D(\gamma) = \top$ and $H'_{\phi'}(\gamma)$ be the corresponding hybrid controller, then for all the trajectories $z(t)$ of Σ' under controller $H'_{\phi'}(\gamma)$ we have $\langle\langle \phi', \mathcal{O}_{\delta}^e \rangle\rangle_C(z) = \top$.*

Theorem 7.4.1 concludes our proposed solution to Problem 7.4.1. The following example illustrates the theoretical results presented in Section 7.4.

Example 7.4.3. *For the construction of the hybrid controller $H'_{\phi'}(\gamma)$ based on the path of Example 7.4.2, we deploy the potential field controllers of Conner et. al. [43] on the cellular decomposition of Fig. 7.4. The resulting trajectory with initial position $(35, 20)$ and velocity bound $\nu = 0.5$ appears in Fig. 7.6.*

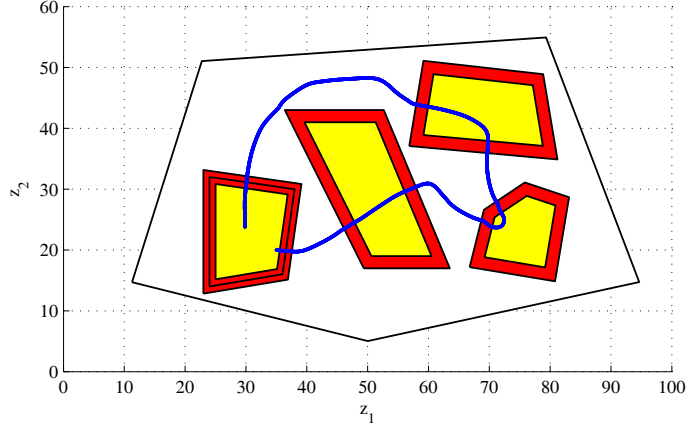


Figure 7.6: A trajectory of system Σ' for the path of Example 7.4.2 using the potential field controllers of [43].

7.5 Putting Everything Together

At this point, we have presented all the pieces that comprise our proposed solution to Problem 7.1.1. Now we are in position to put all the parts together according to the hierarchy proposed in Fig. 7.2. The following theorem, which is immediate from Proposition 7.2.2, Lemma 3.1.2, Corollary 7.3.1 and Theorem 7.4.1, states the main result of this chapter.

Theorem 7.5.1. *Let \mathcal{S}_δ be an approximate simulation relation of precision δ between Σ' and Σ and $u_{\mathcal{S}_\delta}$ be the associated interface. Consider a formula $\phi \in LTL_{\mathbb{B}}(AP)$ and an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and set $\phi' = \mathbf{pos}(\mathbf{nnf}(\phi))$. Let $H'_{\phi'}$ be a controller for Σ' and H_ϕ the associated controller for Σ obtained by interconnection of the elements as shown on Fig. 7.2. Consider some $\varepsilon > \delta$. If for all the trajectories $z(t)$ of Σ' under controller $H'_{\phi'}$ we have $\langle\langle \phi', \mathcal{O}_\varepsilon^e \rangle\rangle_C(z) = \top$, then for all the trajectories $x(t)$ of Σ under controller H_ϕ we have $\langle\langle \phi, \mathcal{O} \rangle\rangle_C(x) = \top$.*

Remark 7.5.1. *Assume that we are given the formula ϕ , the bound μ and the parameter α . It is possible that the maximum allowable value of ν that we can compute from (7.6) renders formula ϕ' unsatisfiable in the modified workspace Z . In this case, we*

can look for a smaller value of ν that will provide a solution to the motion planning problem. The most straightforward solution is to recursively divide ν by 2 until we reach a value where there exists a solution to Problem 7.4.1. Obviously, this procedure might not terminate therefore we need an upper bound on the number of iterations. Note that the more robust the system is with respect to the specification, the faster the robot can go.

Even though our framework regards as input the bound on acceleration μ and then derives the velocity bound ν , in the following examples we give as input the bound ν . We believe that this makes the presentation of the examples clearer.

Example 7.5.1. *The trajectory of system Σ which corresponds to the trajectory of Example 7.4.3 of system Σ' appears in Fig. 7.7. The parameters for this problem are $\nu = 0.5$ and $\alpha = 100$ which implies that μ should at least be 0.5475. Notice that the two trajectories are almost identical since the velocity of Σ' is so low. The total computation time for this example in MATLAB - including the design of the controllers and generation of the trajectories - is about 10 sec. Figure 7.8 shows the modified environment for system Σ' when $\nu = 2.7$, i.e., $\delta = 5.4$. In this case, Problem 7.4.1 does not have a solution, i.e., the formula ψ'_1 is unsatisfiable, since there is no path from ξ_{p_4} back to ξ_{p_1} while remaining in ξ_{-p_2} and ξ_{-p_3} .*

The next example considers larger velocity bounds than Example 7.5.1 and a non-terminating specification.

Example 7.5.2. *Consider the environment in Fig. 7.9 and the LTL formula $\phi = \square(p_0 \wedge \diamond(p_1 \wedge \diamond p_2))$. This specification requires that the robot first visits $\mathcal{O}(p_1)$ and then $\mathcal{O}(p_2)$ repeatedly while always remaining in $\mathcal{O}(p_0)$. For this example, we use the controllers developed in [21] and for the triangulation of the environment we use the*

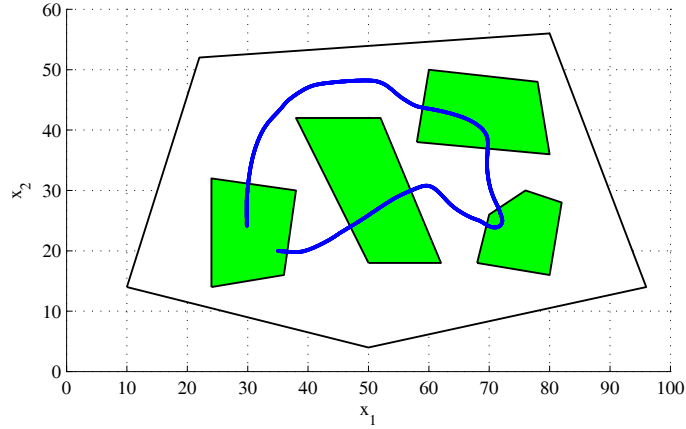


Figure 7.7: The trajectory of system Σ which corresponds to the trajectory of system Σ' presented in Fig. 7.6.

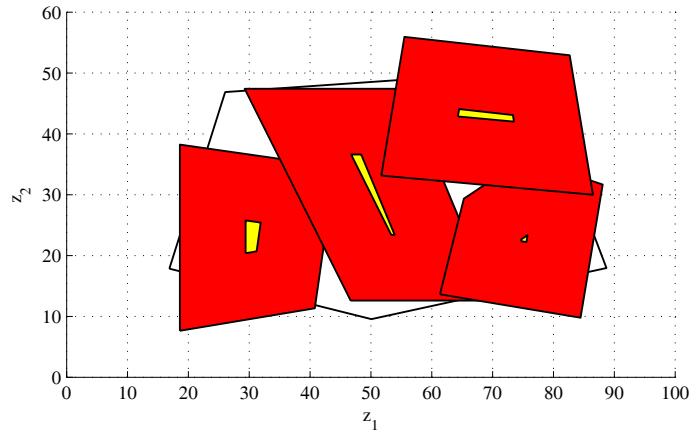


Figure 7.8: Modified workspace for $\nu = 2.7$, i.e., $\delta = 5.4$.

C library [147]. We consider $\nu = 3$ and $a = 100$, therefore, $\delta = 6$. The resulting trajectories appear in Fig. 7.9 and 7.10. The black region in the center of the workspace represents a static obstacle in the environment which is modeled as a hole in $\mathcal{O}(p_0)$. In Fig. 7.11, we present the distance between the trajectories $x(t)$ and $z(t)$. Notice that the distance is always bounded by 6 and that this bound is quite tight.

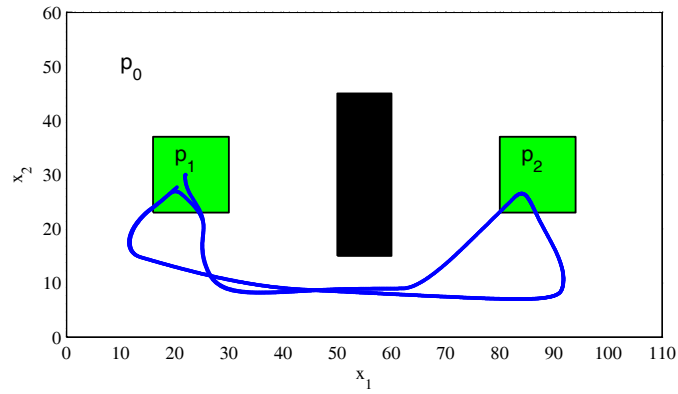


Figure 7.9: The initial environment of Example 7.5.2 and the resulting trajectory $x(t)$ of the dynamic robot Σ .

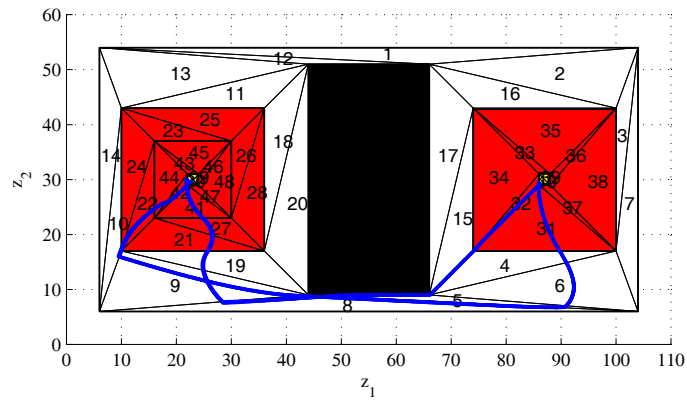


Figure 7.10: The modified environment of Example 7.5.2, its triangulation and the trajectory $z(t)$ of the kinematic model Σ' .

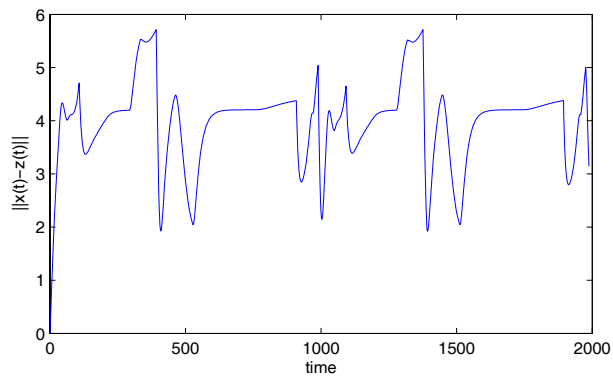


Figure 7.11: The distance between the trajectories $x(t)$ and $z(t)$.

7.6 Related Research

There exist several related approaches to motion planning using hybrid or symbolic methods. For example, the maneuver automata in [68] generate trajectories for helicopters by composing simple dynamic maneuvers. The control quanta [151] solve the navigation problem for non-holonomic vehicles using quantized control. The motion description language [104] and the framework in [115] utilize regular languages in order to guide the construction of hybrid systems. In [112], the author presents a framework for the synthesis of distributed hybrid controllers for an assembly factory given basic controllers and descriptions of the tasks. Finally in [2], the authors present a hierarchical framework for the programming and coordination of robotic groups using the modelling language CHARON.

Our work fundamentally builds upon the concept of sequential composition of controllers [28, 162]. Particularly, we employ methodologies [21, 43, 127] that decompose the workspace or the state space of the robot into convex operational regions and then apply simple controllers in every such region. The advantage of these methods is that they solve the motion planning problem for point robots in complex maze-like environments. However, the deployment of controllers for second order systems [43] is not automatic and it requires user intervention.

The applicability of temporal logics in discrete event systems was advocated as far back as in 1983 [73]. Some of the first explicit applications in robotics appear in [12] and [108]. The first paper deals with the controller synthesis problem for locomotion, while the second with the synchronization of plans for multi-agent systems. In [145], the authors synthesize robust hybrid automata starting from specifications expressed in a modal logic. In [129], generators of models for LTL formulas (Büchi automata) have been utilized as supervisors of multi-robot navigation functions. The

UPPAAL model checking toolbox for timed automata has been applied to the multi-robot motion planning problem in [163], but without taking into account kinematic or dynamic models of the robots. The design of discrete-time controllers that satisfy LTL specifications is addressed in [171]. In [63], controller specifications are derived from a fragment of LTL. These specifications are used to design simple motion controllers that solve the basic path planning problem : “*move from I to the goal G* ”. When these controllers are composed sequentially, the desired motion is generated. More recently, the authors in [114] have demonstrated the applicability of LTL motion planning techniques for swarms of robots building upon their previous work [113].

The work that is the closest related to ours appears in [113]. The authors in [113] extend the framework presented in [61] in order to design hybrid automata with affine dynamics with drift using the controllers presented in [85]. The framework in [113] can also solve Problem 7.1.1, but we advocate that our approach has several clear advantages when one explicitly considers the motion planning problem. First, the hierarchical approach enables the design of control laws for a 2D system instead of a four dimensional one. Second, our approach avoids the state explosion problem introduced by (i) the fine partitioning of the state space with respect to the predicates, and (ii) the consequent tessellation² of the 4D space (see [113]). Finally, the freedom to choose a δ greater than 2ν enables the design of hybrid controllers that can tolerate bounded inaccuracies in the system. For these reasons, we strongly believe that a hierarchical approach can provide a viable solution to a large class of control problems.

²In higher dimensions there do not exist exact space decomposition algorithms and, hence, one has to resort on such approximate partitioning techniques as the tessellation.

7.7 Conclusions and Future Work

We have presented an automatic framework for the solution of the temporal logic motion planning problem for dynamic mobile robots. Our framework is based on hierarchical control, the notion of approximate bisimulation relations and the robustness theory for temporal logic formulas. In the process of building this new framework we have also derived two intermediate results. First, we presented a solution to Problem 7.4.1, i.e., an automatic framework for the solution of the temporal logic motion planning problem for kinematic models. Second, using Theorem 3.1.2, we can construct a more robust solution to Problem 7.4.1, which can account for bounded errors of measure δ in the trajectories of the system. To the best of our knowledge, this thesis presents the first computationally tractable approach to all the above problems.

Future research will concentrate on several directions. First, we are considering employing controllers for nonholonomic systems [42] at the low hierarchical level. Complementary to the first direction, we are investigating new interfaces that can take into account nonholonomic constraints. Another important direction is the extension of this framework to 3D motion planning with application to unmanned aerial vehicles [18]. Finally, we are currently working on converting our single-robot motion planning framework into a reactive multi-robot motion planning system [118].

Part III

Appendix

Chapter 8

Proofs of Part I

8.1 Proofs of Section 3.1

8.1.1 Proof of Theorem 3.1.1

In this proof, we will use the following lemmas.

Lemma 8.1.1. *Let (X, d) be a metric space and $\{S_a\}_{a \in A}$ be an arbitrary collection of subsets of X . For any $x \in X$, $\mathbf{dist}_d(x, \cup_{a \in A} S_a) = \inf_{a \in A} \mathbf{dist}_d(x, S_a)$.*

Proof. For any $x \in X$, we have

$$\begin{aligned} \mathbf{dist}_d(x, \cup_{a \in A} S_a) &= \inf_{y \in \cup_{a \in A} S_a} d(x, y) = \inf_{y \in \cup_{a \in A} S_a} d(x, y) \\ &= \inf_{a \in A} \inf_{y \in S_a} d(x, y) = \inf_{a \in A} \inf_{y \in \overline{S_a}} d(x, y) \\ &= \inf_{a \in A} \mathbf{dist}_d(x, S_a) \end{aligned}$$

□

Lemma 8.1.2. *Let (X, d) be a metric space and $\{S_a\}_{a \in A}$ be an arbitrary collection of subsets of X . For any $x \in X$, $\mathbf{dist}_d(x, \cap_{a \in A} S_a) \geq \sup_{a \in A} \mathbf{dist}_d(x, S_a)$.*

Proof. We have that $\bigcap_{a \in A} S_a \subseteq S_a$ for any $a \in A$. Therefore, $\mathbf{dist}_d(x, \bigcap_{a \in A} S_a) \geq \mathbf{dist}_d(x, S_a)$. Since this holds for any $a \in A$ we get that $\mathbf{dist}_d(x, \bigcap_{a \in A} S_a) \geq \sup_{a \in A} \mathbf{dist}_d(x, S_a)$. \square

Lemma 8.1.3. *Consider an atomic proposition $p \in AP$, an observation map $\mathcal{O} \in \mathfrak{F}(AP, \mathcal{P}(X))$ and a continuous-time signal $s \in \mathfrak{F}(R, X)$, then for any time $t \in R$, we have $\mathbf{Dist}_\rho(s, \mathcal{L}_t(p)) = \llbracket p \rrbracket_C(s, t)$.*

Proof. We only show the proof for the case that $s \in \mathcal{L}_t(p)$, because the proof for the case $s \notin \mathcal{L}_t(p)$ is similar. We have

$$\begin{aligned} \mathbf{Dist}_\rho(s, \mathcal{L}_t(p)) &= \mathbf{depth}_\rho(s, \mathcal{L}_t(p)) = \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg p)) \\ &= \inf_{s' \in \overline{\mathcal{L}_t(\neg p)}} \rho(s, s') = \inf_{s' \in \overline{\mathcal{L}_t(\neg p)}} \sup_{t' \in R} d(s(t'), s'(t')) \\ &= \inf_{s' \in \overline{\mathcal{L}_t(\neg p)}} \max\{d(s(t), s'(t)), \sup_{t' \in R_{\neq t}} d(s(t'), s'(t'))\} \end{aligned}$$

For each $s' \in \overline{\mathcal{L}_t(\neg p)}$ with $d(s(t), s'(t)) < \sup_{t' \in R_{\neq t}} d(s(t'), s'(t'))$, there exists some $s'' \in \overline{\mathcal{L}_t(\neg p)}$ with $s''(t) = s'(t)$ and $s''(t') = s(t')$ for all $t' \in R_{\neq t}$. That is, $0 = \sup_{t' \in R_{\neq t}} d(s(t'), s''(t')) \leq d(s(t), s''(t)) = d(s(t), s'(t)) < \sup_{t' \in R_{\neq t}} d(s(t'), s'(t'))$ or in other words $\rho(s, s'') < \rho(s, s')$. Thus,

$$\begin{aligned} \mathbf{Dist}_\rho(s, \mathcal{L}_t(p)) &= \inf_{s' \in \overline{\mathcal{L}_t(\neg p)}} d(s(t), s'(t)) = \inf_{x \in \overline{X \setminus \mathcal{O}(p)}} d(s(t), x) \\ &= \mathbf{dist}_d(s(t), X \setminus \mathcal{O}(p)) = \llbracket p \rrbracket_C(s, t) \end{aligned}$$

\square

The proof of Theorem 3.1.1 is by induction on the structure of formula ϕ .

Constant $\phi = \top$: We have $\mathcal{L}_t(\top) = \mathfrak{F}(R, X)$, thus

$$0 = -\mathbf{dist}_\rho(s, \mathcal{L}_t(\top)) \leq \llbracket \top \rrbracket_C(s, t) = +\infty = \mathbf{dist}_\rho(s, \emptyset) = \mathbf{depth}_\rho(s, \mathcal{L}_t(\top))$$

Atomic Propositions $\phi = p$: Immediate from Lemma 8.1.3.

Negation $\phi = \neg\phi_1$: By the induction hypothesis, we have

$$\begin{aligned} -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_1)) &\leq \llbracket \phi_1 \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_1)) \implies \\ -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_1)) &\leq -\llbracket \neg\phi_1 \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_1)) \implies \\ \mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_1)) &\geq \llbracket \neg\phi_1 \rrbracket_C(s, t) \geq -\mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_1)) \implies \\ \mathbf{depth}_\rho(s, \mathcal{L}_t(\neg\phi_1)) &\geq \llbracket \neg\phi_1 \rrbracket_C(s, t) \geq -\mathbf{dist}_\rho(s, \mathcal{L}_t(\neg\phi_1)) \implies \\ -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi)) &\leq \llbracket \phi \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi)) \end{aligned}$$

Disjunction $\phi = \phi_1 \vee \phi_2$: By the induction hypothesis we get that for $i = 1, 2$

$$-\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_i)) \leq \llbracket \phi_i \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_i))$$

for $i = 1, 2$. Thus, by the monotonicity property of the supremum, we get

$$\bigsqcup_{i=1,2} -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_i)) \leq \bigsqcup_{i=1,2} \llbracket \phi_i \rrbracket_C(s, t) \leq \bigsqcup_{i=1,2} \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_i))$$

Note that by the definition of the language we get

$$\mathcal{L}_t(\phi) = \mathcal{L}_t(\phi_1 \vee \phi_2) = \mathcal{L}_t(\phi_1) \cup \mathcal{L}_t(\phi_2) \tag{8.1}$$

Moreover, by eq. (8.1) and Lemma 8.1.1, we have

$$\begin{aligned} \bigsqcup_{i=1,2} -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_i)) &= - \prod_{i=1,2} \mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_i)) = \\ &= -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi_1) \cup \mathcal{L}_t(\phi_2)) = -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi)) \end{aligned}$$

Also, by eq. (8.1) and Lemma 8.1.2, we have

$$\begin{aligned} \bigsqcup_{i=1,2} \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi_i)) &= \bigsqcup_{i=1,2} \mathbf{dist}_\rho(s, \mathfrak{F}(R, X) \setminus \mathcal{L}_t(\phi_i)) \leq \\ &\leq \mathbf{dist}_\rho(s, \bigcap_{i=1,2} \mathfrak{F}(R, X) \setminus \mathcal{L}_t(\phi_i)) = \mathbf{dist}_\rho(s, \mathfrak{F}(R, X) \setminus \bigcup_{i=1,2} \mathcal{L}_t(\phi_i)) = \\ &= \mathbf{depth}_\rho(s, \bigcup_{i=1,2} \mathcal{L}_t(\phi_i)) = \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi)) \end{aligned}$$

Thus, by definition we have

$$-\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi)) \leq \llbracket \phi \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi))$$

Until $\phi = \phi_1 \mathcal{U}_I \phi_2$: By definition, we have

$$\llbracket \phi_1 \mathcal{U}_I \phi_2 \rrbracket_C(s, t) = \bigsqcup_{t' \in (t +_R \mathcal{I})} (\llbracket \phi_2 \rrbracket_C(s, t') \sqcap \prod_{t < t'' < t'} \llbracket \phi_1 \rrbracket_C(s, t''))$$

By the induction hypothesis, we get

$$-\mathbf{dist}_\rho(s, \mathcal{L}_{t'}(\phi_2)) \leq \llbracket \phi_2 \rrbracket_C(s, t') \leq \mathbf{depth}_\rho(s, \mathcal{L}_{t'}(\phi_2))$$

for any $t' \in (t +_R \mathcal{I})$ and

$$-\mathbf{dist}_\rho(s, \mathcal{L}_{t''}(\phi_1)) \leq \llbracket \phi_1 \rrbracket_C(s, t'') \leq \mathbf{depth}_\rho(s, \mathcal{L}_{t''}(\phi_1))$$

for any $t'' \in (t, t')$. By the monotonicity property of infimum we have

$$\prod_{t'' \in (t, t')} -\mathbf{dist}_\rho(s, \mathcal{L}_{t''}(\phi_1)) \leq \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \prod_{t'' \in (t, t')} \mathbf{depth}_\rho(s, \mathcal{L}_{t''}(\phi_1)).$$

Also, by Lemma 8.1.2 we have that

$$\begin{aligned} \mathbf{dist}_\rho(s, \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1)) &\geq \bigsqcup_{t'' \in (t, t')} \mathbf{dist}_\rho(s, \mathcal{L}_{t''}(\phi_1)) \implies \\ -\mathbf{dist}_\rho(s, \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1)) &\leq \prod_{t'' \in (t, t')} -\mathbf{dist}_\rho(s, \mathcal{L}_{t''}(\phi_1)) \end{aligned}$$

and by Lemma 8.1.1 we have that

$$\begin{aligned} \prod_{t'' \in (t, t')} \mathbf{depth}_\rho(s, \mathcal{L}_{t''}(\phi_1)) &= \prod_{t'' \in (t, t')} \mathbf{dist}_\rho(s, \mathfrak{F}(R, X) \setminus \mathcal{L}_{t''}(\phi_1)) = \\ &= \mathbf{dist}_\rho(s, \bigcup_{t'' \in (t, t')} \mathfrak{F}(R, X) \setminus \mathcal{L}_{t''}(\phi_1)) = \\ &= \mathbf{dist}_\rho(s, \mathfrak{F}(R, X) \setminus \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1)) = \mathbf{depth}_\rho(s, \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1)). \end{aligned}$$

Therefore, we have that

$$-\mathbf{dist}_\rho \left(s, \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1) \right) \leq \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \mathbf{depth}_\rho \left(s, \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1) \right).$$

Similarly, we get that for any $t' \in (t +_R \mathcal{I})$

$$\begin{aligned} -\mathbf{dist}_\rho \left(s, \mathcal{L}_{t'}(\phi_2) \cap \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1) \right) &\leq \llbracket \phi_2 \rrbracket_C(s, t') \sqcap \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \\ &\leq \mathbf{depth}_\rho \left(s, \mathcal{L}_{t'}(\phi_2) \cap \bigcap_{t'' \in (t, t')} \mathcal{L}_{t''}(\phi_1) \right). \end{aligned}$$

Finally, similar to the disjunction case, we get that

$$\begin{aligned}
& -\mathbf{dist}_\rho \left(s, \bigcup_{t' \in (t+R\mathcal{I})} \left(\mathcal{L}_{t'}(\phi_2) \cap \bigcap_{t'' \in (t,t')} \mathcal{L}_{t''}(\phi_1) \right) \right) \leq \\
& \bigsqcup_{t' \in (t+R\mathcal{I})} \left(\llbracket \phi_2 \rrbracket_C(s, t') \sqcap \prod_{t'' \in (t,t')} \llbracket \phi_1 \rrbracket_C(s, t'') \right) \leq \\
& \leq \mathbf{depth}_\rho \left(s, \bigcup_{t' \in (t+R\mathcal{I})} \left(\mathcal{L}_{t'}(\phi_2) \cap \bigcap_{t'' \in (t,t')} \mathcal{L}_{t''}(\phi_1) \right) \right).
\end{aligned}$$

Since

$$\begin{aligned}
\mathcal{L}_t(\phi) &= \{s \in \mathfrak{F}(R, X) \mid \langle\langle \phi \rangle\rangle_C(s, t) = \top\} = \\
&= \left\{ s \in \mathfrak{F}(R, X) \mid \bigsqcup_{t' \in (t+R\mathcal{I})} (\langle\langle \phi_2 \rangle\rangle_C(s, t') \sqcap \prod_{t < t'' < t'} \langle\langle \phi_1 \rangle\rangle_C(s, t'')) = \top \right\} = \\
&= \left\{ s \in \mathfrak{F}(R, X) \mid \bigvee_{t' \in (t+R\mathcal{I})} (\langle\langle \phi_2 \rangle\rangle_C(s, t') = \top \wedge \bigwedge_{t < t'' < t'} \langle\langle \phi_1 \rangle\rangle_C(s, t'') = \top) \right\} = \\
&= \bigcup_{t' \in (t+R\mathcal{I})} \left(\mathcal{L}_{t'}(\phi_2) \cap \bigcap_{t'' \in (t,t')} \mathcal{L}_{t''}(\phi_1) \right),
\end{aligned}$$

we conclude that $-\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi)) \leq \llbracket \phi \rrbracket_C(s, t) \leq \mathbf{depth}_\rho(s, \mathcal{L}_t(\phi))$.

8.1.2 Proof of Proposition 3.1.2

Note that the first statement – i.e., if $\llbracket \phi \rrbracket_C(s, t) > 0$, then $\langle\langle \phi \rangle\rangle_C(s, t) = \top$, and if $\llbracket \phi \rrbracket_C(s, t) < 0$, then $\langle\langle \phi \rangle\rangle_C(s, t) = \perp$ – is immediate from Corollary 3.1.1. Therefore, we will only prove by induction on the structure of the formula $\phi \in MTL_{\mathbb{B}}$ the second statement, that is, if $\langle\langle \phi \rangle\rangle_C(s, t) = \top$, then $\llbracket \phi \rrbracket_C(s, t) \geq 0$, and if $\langle\langle \phi \rangle\rangle_C(s, t) = \perp$, then $\llbracket \phi \rrbracket_C(s, t) \leq 0$.

Case $\phi = \top$ or $\phi = \perp$: Immediate from the semantics.

Case $\phi = p \in AP$: If $\langle\langle\phi\rangle\rangle_C(s, t) = \top$, then by definition $s(t) \in \mathcal{O}(p)$, which implies that $\mathbf{Dist}_a(s(t), \mathcal{O}(p)) \geq 0$, and, thus, that $\llbracket\phi\rrbracket_C(s, t) \geq 0$. If on the other hand $\langle\langle\phi\rangle\rangle_C(s, t) = \perp$, then by definition $s(t) \notin \mathcal{O}(p)$, which implies that $\mathbf{Dist}_a(s(t), \mathcal{O}(p)) \leq 0$ and, thus, that $\llbracket\phi\rrbracket_C(s, t) \leq 0$.

Case $\phi = \neg\phi_1$: **(i)** If $\langle\langle\phi\rangle\rangle_C(s, t) = \top$, then by definition $\langle\langle\phi_1\rangle\rangle_C(s, t) = \perp$. By the induction hypothesis, we get that $\llbracket\phi_1\rrbracket_C(s, t) \leq 0$, which implies $\llbracket\neg\phi_1\rrbracket_C(s, t) \geq 0$. **(ii)** If $\langle\langle\phi\rangle\rangle_C(s, t) = \perp$, then by definition $\langle\langle\phi_1\rangle\rangle_C(s, t) = \top$. By the induction hypothesis, we get that $\llbracket\phi_1\rrbracket_C(s, t) \geq 0$, which implies $\llbracket\neg\phi_1\rrbracket_C(s, t) \leq 0$.

Case $\phi = \phi_1 \vee \phi_2$: **(i)** If $\langle\langle\phi_1 \vee \phi_2\rangle\rangle_C(s, t) = \top$, then by definition we get that $\langle\langle\phi_1\rangle\rangle_C(s, t) = \top$ or $\langle\langle\phi_2\rangle\rangle_C(s, t) = \top$. By the induction hypothesis, we have $\llbracket\phi_1\rrbracket_C(s, t) \geq 0$ or $\llbracket\phi_2\rrbracket_C(s, t) \geq 0$. Thus, $\llbracket\phi_1\rrbracket_C(s, t) \sqcup \llbracket\phi_2\rrbracket_C(s, t) \geq 0$, which implies $\llbracket\phi\rrbracket_C(s, t) \geq 0$. **(ii)** If $\langle\langle\phi_1 \vee \phi_2\rangle\rangle_C(s, t) = \perp$, then by definition $\langle\langle\phi_1\rangle\rangle_C(s, t) = \perp$ and $\langle\langle\phi_2\rangle\rangle_C(s, t) = \perp$. By the induction hypothesis, we get that $\llbracket\phi_1\rrbracket_C(s, t) \leq 0$ and $\llbracket\phi_2\rrbracket_C(s, t) \leq 0$. Thus, $\llbracket\phi_1\rrbracket_C(s, t) \sqcup \llbracket\phi_2\rrbracket_C(s, t) \leq 0$, which implies $\llbracket\phi\rrbracket_C(s, t) \leq 0$.

Case $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: **(i)** If $\langle\langle\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2\rangle\rangle_C(s, t) = \top$, then by the definition of until, there exists some time $t' \in (t +_R \mathcal{I})$ such that $\langle\langle\phi_2\rangle\rangle_C(s, t') = \top$ and for all $t'' \in (t, t')$, we have $\langle\langle\phi_1\rangle\rangle_C(s, t'') = \top$. Using the induction hypothesis we get that $\llbracket\phi_2\rrbracket_C(s, t') \geq 0$ and for all $t'' \in (t, t')$, we have $\llbracket\phi_1\rrbracket_C(s, t'') \geq 0$. Therefore, $\llbracket\phi\rrbracket_C(s, t) = \bigsqcup_{t' \in (t +_R \mathcal{I})} (\llbracket\phi_2\rrbracket_C(s, t') \sqcap \bigsqcap_{t < t'' < t'} \llbracket\phi_1\rrbracket_C(s, t'')) \geq 0$. **(ii)** If $\langle\langle\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2\rangle\rangle_C(s, t) = \perp$, then $(t +_R \mathcal{I}) = \emptyset$ or for all time $t' \in (t +_R \mathcal{I})$, we have $\langle\langle\phi_2\rangle\rangle_C(s, t') \sqcap \bigsqcap_{t < t'' < t'} \langle\langle\phi_1\rangle\rangle_C(s, t'') = \perp$. In the former case, we immediately get by the definition that $\llbracket\phi\rrbracket_C(s, t) = -\infty$. In the latter case, for all time $t' \in (t +_R \mathcal{I})$, we have $\langle\langle\phi_2\rangle\rangle_C(s, t') = \perp$ or there exists some time $t'' \in (t, t')$ such that $\langle\langle\phi_1\rangle\rangle_C(s, t'') = \perp$. Using the induction hypothesis we get that for all $t' \in (t +_R \mathcal{I})$, $\llbracket\phi_2\rrbracket_C(s, t') \leq 0$ or there exists $t'' \in (t, t')$ such that $\llbracket\phi_1\rrbracket_C(s, t'') \leq 0$. Therefore, $\llbracket\phi\rrbracket_C(s, t) \leq 0$ by definition.

8.1.3 Proof of Theorem 3.1.2

We prove by induction on the structure of $\phi' = \mathbf{pos}(\mathbf{nnf}(\phi))$ that $\langle\langle\phi', \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ implies $\llbracket\phi', \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$. Then, the result follows by Lemma 3.1.2.

Case $\phi' = \xi \in \Xi_{AP}$: We have $\langle\langle\xi, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ iff $s(t) \in \mathcal{O}_\varepsilon^e(\xi) = C_d(\mathcal{O}^e(\xi), \varepsilon)$. The later implies that $\overline{B_d(s(t), \varepsilon)} \subseteq \mathcal{O}^e(\xi)$ or $\llbracket\xi, \mathcal{O}^e\rrbracket_C(s, t) = \mathbf{Dist}_d(s(t), \mathcal{O}^e(\xi)) = \mathbf{depth}_d(s(t), \mathcal{O}^e(\xi)) \geq \varepsilon$.

Case $\phi' = \phi_1 \vee \phi_2$: $\langle\langle\phi', \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ or by definition $\langle\langle\phi_1, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ or $\langle\langle\phi_2, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$. Using the induction hypothesis, we get that $\llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$ or $\llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$. That is, $\llbracket\phi', \mathcal{O}^e\rrbracket_C(s, t) = \llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t) \sqcup \llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon \sqcup \varepsilon = \varepsilon$.

Case $\phi' = \phi_1 \wedge \phi_2$: $\langle\langle\phi', \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ or by definition $\langle\langle\phi_1, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ and $\langle\langle\phi_2, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$. Using the induction hypothesis, we get that $\llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$ and $\llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$. That is, $\llbracket\phi', \mathcal{O}^e\rrbracket_C(s, t) = \llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t) \sqcap \llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon \sqcap \varepsilon = \varepsilon$.

Case $\phi' = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: We have $\langle\langle\phi', \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ iff by definition there exists $t' \in (t +_R \mathcal{I})$ such that $\langle\langle\phi_2, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t') = \top$ and for all $t'' \in (t, t')$ we have $\langle\langle\phi_1, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t'') = \top$. By the induction hypothesis, we get that $\llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t') \geq \varepsilon$ and that for all $t'' \in (t, t')$, $\llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t'') \geq \varepsilon$. Therefore, $\llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t') \sqcap \prod_{t < t'' < t'} \llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t'') \geq \varepsilon \sqcap \prod_{t < t'' < t'} \varepsilon = \varepsilon$ and, thus,

$$\bigsqcup_{t' \in (t +_R \mathcal{I})} \left(\llbracket\phi_2, \mathcal{O}^e\rrbracket_C(s, t') \sqcap \prod_{t < t'' < t'} \llbracket\phi_1, \mathcal{O}^e\rrbracket_C(s, t'') \right) \geq \varepsilon$$

or $\llbracket\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2, \mathcal{O}^e\rrbracket_C(s, t) \geq \varepsilon$.

Case $\phi' = \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2$: We have $\langle\langle\phi', \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t) = \top$ iff by definition either $(t +_R \mathcal{I}) = \emptyset$ or for all $t' \in (t +_R \mathcal{I})$ we have $\langle\langle\phi_2, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t') = \top$ or there exists some $t'' \in (t, t')$ such that $\langle\langle\phi_1, \mathcal{O}_\varepsilon^e\rangle\rangle_C(s, t'') = \top$. In the first case, we get $\llbracket\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2, \mathcal{O}^e\rrbracket_C(s, t) = +\infty$ by

definition. In the latter case, by the induction hypothesis, we get that for all $t' \in (t +_R \mathcal{I})$ we have $\llbracket \phi_2, \mathcal{O}^e \rrbracket_C(s, t') \geq \varepsilon$ or there exists $t'' \in (t, t')$ such that $\llbracket \phi_1, \mathcal{O}^e \rrbracket_C(s, t'') \geq \varepsilon$. Therefore, for all $t' \in (t +_R \mathcal{I})$ we have $\llbracket \phi_2, \mathcal{O}^e \rrbracket_C(s, t') \sqcup \bigsqcup_{t < t'' < t'} \llbracket \phi_1, \mathcal{O}^e \rrbracket_C(s, t'') \geq \varepsilon$ and, thus,

$$\prod_{t' \in (t +_R \mathcal{I})} \left(\llbracket \phi_2, \mathcal{O}^e \rrbracket_C(s, t') \sqcup \bigsqcup_{t < t'' < t'} \llbracket \phi_1, \mathcal{O}^e \rrbracket_C(s, t'') \right) \geq \varepsilon$$

or $\llbracket \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2, \mathcal{O}^e \rrbracket_C(s, t) \geq \varepsilon$.

8.1.4 Proof of Proposition 3.1.3

The proof of Proposition 3.1.3 is by induction on the structure of ϕ .

Constant $\phi = \top$: We have

$$\mathbf{Dist}_\rho(s, \mathcal{L}_t(\top)) = \mathbf{depth}_\rho(s, \mathcal{L}_t(\top)) = \mathbf{dist}_\rho(s, \emptyset) = +\infty = \llbracket \top \rrbracket_C(s, t)$$

Atomic Propositions $\phi = p$ or $\phi = \neg p$ with $p \in AP$: Immediate from Lemma 8.1.3.

Conjunction $\phi = \phi_1 \wedge \phi_2$: Since $\langle\langle \phi \rangle\rangle_C(s, t) = \top$, we have $\langle\langle \phi_1 \rangle\rangle_C(s, t) = \top$ and $\langle\langle \phi_2 \rangle\rangle_C(s, t) = \top$. By the induction hypothesis we get that $\llbracket \phi_1 \rrbracket_C(s, t) = \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi_1))$ and $\llbracket \phi_2 \rrbracket_C(s, t) = \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi_2))$. Moreover,

$$\mathcal{L}_t(\neg \phi) = \mathcal{L}_t(\neg \phi_1 \vee \neg \phi_2) = \mathcal{L}_t(\neg \phi_1) \cup \mathcal{L}_t(\neg \phi_2).$$

Hence, using Lemma 8.1.1, and the induction hypothesis we have

$$\begin{aligned} \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi)) &= \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi)) = \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi_1) \cup \mathcal{L}_t(\neg \phi_2)) \\ &= \min\{\mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi_1)), \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg \phi_2))\} \end{aligned}$$

$$= \llbracket \phi_1 \rrbracket_C(s, t) \sqcap \llbracket \phi_2 \rrbracket_C(s, t) = \llbracket \phi \rrbracket_C(s, t)$$

Always $\phi = \Box_{\mathcal{I}}\phi_1$: Since $\langle\langle \phi \rangle\rangle_C(s, t) = \top$, we get that $(t +_R \mathcal{I}) = \emptyset$ or that for all $t' \in (t +_R \mathcal{I})$, we have $\langle\langle \phi_1 \rangle\rangle_C(s, t') = \top$. In the former case, we immediately get that $\mathcal{L}_t(\phi) = \mathfrak{F}(R, X)$ and

$$\mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi)) = \mathbf{dist}_\rho(s, \emptyset) = +\infty = \sqcap_{t' \in \emptyset} \llbracket \phi_1 \rrbracket_C(s, t') = \llbracket \phi \rrbracket_C(s, t)$$

In the latter case, by the induction hypothesis we get that for all $t' \in (t +_R \mathcal{I})$, we have $\llbracket \phi_1 \rrbracket_C(s, t') = \mathbf{dist}_\rho(s, \mathcal{L}_{t'}(\neg\phi_1))$. Also, $\mathcal{L}_t(\neg\phi) = \cup_{t' \in (t +_R \mathcal{I})} \mathcal{L}_{t'}(\neg\phi_1)$. Hence, using Lemma 8.1.1, and the induction hypothesis we have

$$\begin{aligned} \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi)) &= \mathbf{dist}_\rho(s, \mathcal{L}_t(\neg\phi)) = \mathbf{dist}_\rho(s, \cup_{t' \in (t +_R \mathcal{I})} \mathcal{L}_{t'}(\neg\phi_1)) \\ &= \inf_{t' \in (t +_R \mathcal{I})} \mathbf{dist}_\rho(s, \mathcal{L}_{t'}(\neg\phi_1)) = \sqcap_{t' \in (t +_R \mathcal{I})} \llbracket \phi_1 \rrbracket_C(s, t') \\ &= \llbracket \phi \rrbracket_C(s, t) \end{aligned}$$

8.1.5 Proof of Corollary 3.1.2

Note that if $\phi \in MTL_{\mathbb{B}}(\vee, \diamond)$, then $\psi = \mathbf{nnf}(\neg\phi) \in MTL_{\mathbb{B}}(\wedge, \square)$. Also, since $\langle\langle \phi \rangle\rangle_C(s, t) = \perp$, we have $\langle\langle \psi \rangle\rangle_C(s, t) = \top$. Then, by Proposition 3.1.3, we have

$$\begin{aligned} \llbracket \phi \rrbracket_C(s, t) &= -\llbracket \neg\phi \rrbracket_C(s, t) = -\llbracket \psi \rrbracket_C(s, t) \\ &= -\mathbf{dist}_\rho(s, \mathcal{L}_t(\neg\psi)) = -\mathbf{dist}_\rho(s, \mathcal{L}_t(\phi)) = \mathbf{Dist}_\rho(s, \mathcal{L}_t(\phi)) \end{aligned}$$

8.2 Proofs of Section 3.2

Most of the proofs in Section 3.2 are essentially identical to the proofs of Section 2 and, hence, we omit these proofs.

8.2.1 Recursive Formulation of Until in Section 3.2.3

Starting from the definition of until, we have

$$\begin{aligned}
\llbracket \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rrbracket_D(\mu, i) &= \bigsqcup_{j \in \tau^{-1}(\tau(i) + \mathcal{I})} (\llbracket \phi_2 \rrbracket_D(\mu, j) \sqcap \prod_{i \leq k < j} \llbracket \phi_1 \rrbracket_D(\mu, k)) \\
&= \bigsqcup_{j \geq i} (K_{\epsilon}^{\infty}(\tau(j), \tau(i) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, j) \sqcap \prod_{i \leq k < j} \llbracket \phi_1 \rrbracket_D(\mu, k)) \\
&= (K_{\epsilon}^{\infty}(\tau(i), \tau(i) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i)) \sqcup \\
&\quad \sqcup \bigsqcup_{j \geq i+1} (K_{\epsilon}^{\infty}(\tau(j), \tau(i) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, j) \sqcap \prod_{i \leq k < j} \llbracket \phi_1 \rrbracket_D(\mu, k)) \\
&= (K_{\epsilon}^{\infty}(\tau(i), \tau(i) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i)) \sqcup (\llbracket \phi_1 \rrbracket_D(\mu, i) \sqcap \\
&\quad \sqcap \bigsqcup_{j \geq i+1} (K_{\epsilon}^{\infty}(\tau(j), (\tau(i+1) - \delta\tau(i)) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, j) \sqcap \\
&\quad \sqcap \prod_{i+1 \leq k < j} \llbracket \phi_1 \rrbracket_D(\mu, k))) \\
&= (K_{\epsilon}^{\infty}(\tau(i), \tau(i) +_R \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i)) \sqcup \\
&\quad \sqcup (\llbracket \phi_1 \rrbracket_D(\mu, i) \sqcap \llbracket \phi_1 \mathcal{U}_{(-\delta\tau(i)) +_R \mathcal{I}} \phi_2 \rrbracket_D(\mu, i+1))
\end{aligned}$$

Since $\tau(i) \in R$ for all $i \in N$, we have $\tau(i) \in (\tau(i) +_R \mathcal{I})$ iff $\tau(i) \in (\tau(i) + \mathcal{I})$ iff $0 \in \mathcal{I}$. Also, when N is finite and $i = \max N$, we have $\tau(i+1) +_R \mathcal{I} = \emptyset$. Hence, $\llbracket \phi_1 \mathcal{U}_{(-\delta\tau(i)) +_R \mathcal{I}} \phi_2 \rrbracket_D(\mu, i+1) = -\infty$. Thus, we derive the recursive formulation of until in Section 3.2.3.

8.2.2 Proof of Lemma 3.2.2

The proof uses induction on the structure of ϕ . Since $i < \max N$, we have $last = \perp$.

Base case $\phi = p \in AP$: then

$$\begin{aligned} \llbracket p \rrbracket_D(\mu, i) &= \mathbf{Dist}_d(\sigma(i), \mathcal{O}(p)) = \mathbf{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \\ &= \llbracket \mathbf{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \end{aligned}$$

since $\mathbf{Dist}_d(\sigma(i), \mathcal{O}(p)) \in \overline{\mathbb{R}}$. The proof is similar when $\phi = \top$ or $\phi = c \in \overline{\mathbb{R}}$.

Negation $\phi = \neg\phi_1$: then

$$\begin{aligned} \llbracket \phi \rrbracket_D(\mu, i) &= -\llbracket \phi_1 \rrbracket_D(\mu, i) \\ &= -\llbracket \mathbf{DERIVE}(\phi_1, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \quad \text{by I.H.} \\ &= \llbracket \neg\mathbf{DERIVE}(\phi_1, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \\ &= \llbracket \mathbf{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \end{aligned}$$

Disjunction $\phi = \phi_1 \vee \phi_2$: then

$$\begin{aligned} \llbracket \phi \rrbracket_D(\mu, i) &= \llbracket \phi_1 \rrbracket_D(\mu, i) \sqcup \llbracket \phi_2 \rrbracket_D(\mu, i) \\ &= \llbracket \mathbf{DERIVE}(\phi_1, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \sqcup \quad \text{by I.H.} \\ &\quad \sqcup \llbracket \mathbf{DERIVE}(\phi_2, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \\ &= \llbracket \mathbf{DERIVE}(\phi_1, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \vee \\ &\quad \vee \mathbf{DERIVE}(\phi_2, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \\ &= \llbracket \mathbf{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \end{aligned}$$

Until $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: Now using the equivalence from Section 8.2.1, we derive

$$\begin{aligned}
\llbracket \phi \rrbracket_D(\mu, i) &= (K_{\varepsilon}^{\infty}(0, \mathcal{I}) \sqcap \llbracket \phi_2 \rrbracket_D(\mu, i)) \sqcup \\
&\quad \sqcup (\llbracket \phi_1 \rrbracket_D(\mu, i) \sqcap \llbracket \phi_1 \mathcal{U}_{(-\delta\tau(i)) + R\mathcal{I}} \phi_2 \rrbracket_D(\mu, i + 1)) \\
&= (K_{\varepsilon}^{\infty}(0, \mathcal{I}) \sqcap \llbracket \text{DERIVE}(\phi_2, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1)) \sqcup \\
&\quad \sqcup (\llbracket \text{DERIVE}(\phi_1, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1) \sqcap \\
&\quad \sqcap \llbracket \phi_1 \mathcal{U}_{(-\delta\tau(i)) + R\mathcal{I}} \phi_2 \rrbracket_D(\mu, i + 1)) \\
&= \llbracket \text{DERIVE}(\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}) \rrbracket_D(\mu, i + 1)
\end{aligned}$$

by I.H.

8.3 Proofs of Section 4.2

8.3.1 Proof of Theorem 4.2.3

Fix a point $p_1 \in P_1$ and a point $p_2 \in P_2$. Then, by Corollary 4.2.1, $\mathcal{F}(x_1, x_2) = \sqrt{\mathcal{V}(x_1, x_2)}$ is a bisimulation function between the systems

$$\bar{\Sigma}_{1'} = (R, X_1, X_1^0, Y, f_1(\cdot, p_1), g_1)$$

and

$$\bar{\Sigma}_{2'} = (R, X_2, X_2^0, Y, f_2(\cdot, p_2), g_2).$$

Since this holds for any $p_1 \in P_1$ and any $p_2 \in P_2$ we derive the desired result.

Chapter 9

Proofs of Part II

9.1 Proofs of Chapter 5

9.1.1 Proof of Proposition 5.3.1

The proof is by induction on the structure of formula ϕ . The cases for \top , p and $\neg p$ are immediate from the definitions. Similarly, the cases $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$ are immediate from the induction hypothesis.

Case $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: We have $\langle\langle \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{U}_{C_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rangle\rangle_D(\mu, i) = \top$. Thus, by definition and the induction hypothesis, there exists some

$$j \in \hat{\tau}^{-1}(\hat{\tau}(i) + C_{d_1}(\mathcal{I}, \Delta\hat{\tau})) \subseteq \hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I})$$

such that $\langle\langle \phi_2 \rangle\rangle_D(\mu, j) = \top$ and for all k such that $i \leq k < j$ we have $\langle\langle \phi_1 \rangle\rangle_D(\mu, k) = \top$.

We conclude that $\langle\langle \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rangle\rangle_D(\mu, i) = \top$ by the definition of until. Note that if $C_{d_1}(\mathcal{I}, \Delta\hat{\tau}) = \emptyset$, then $\mathbf{str}_{\Delta\hat{\tau}}(\phi)$ would evaluate to \perp which is a contradiction.

Case $\phi = \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2$: We have $\langle\langle \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{R}_{E_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rangle\rangle_D(\mu, i) = \top$. Thus, by definition and the induction hypothesis, for all $j \in \hat{\tau}^{-1}(\hat{\tau}(i) + E(\mathcal{I}, \Delta\hat{\tau}))$, we

have $\langle\langle\phi_2\rangle\rangle_D(\mu, j) = \top$ or there exists $k \in [i, j)$ such that $\langle\langle\phi_1\rangle\rangle_D(\mu, k) = \top$. Since $\hat{\tau}^{-1}(\hat{\tau}(i) + E_{d_1}(\mathcal{I}, \Delta\hat{\tau})) \supseteq \hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I})$, by the definition of release, we conclude that $\langle\langle\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2\rangle\rangle_D(\mu, i) = \top$.

9.1.2 Proof of Lemma 5.3.1

If both R and \mathcal{I} are unbounded, then we immediately get $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}) \neq \emptyset$ since $\hat{\tau}$ is strictly increasing and diverging. Assume now that \mathcal{I} is bounded and that for some $i \in I$ we get that $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}) = \emptyset$. In other words, we assume that for all $j \geq i$ (since $\hat{\tau}$ is strictly increasing), we have $\hat{\tau}(j) \notin (\hat{\tau}(i) + \mathcal{I})$. Then, one of the following two options must hold since $\hat{\tau}(i) + \mathcal{I}$ is an interval of R :

1. All the samples $j \geq i$ map to points in time that occur sooner than the minimum required time by $\hat{\tau}(i) + \mathcal{I}$. Formally, for all $j \in N_{\geq i}$ we have $\hat{\tau}(j) \prec \inf(\hat{\tau}(i) + \mathcal{I})$, where $\prec \in \{<, \leq\}$ depending on the bounds of \mathcal{I} . Note that this can only be the case when R is bounded, i.e., N is bounded and, thus, $\hat{\tau}(\max N) \prec \inf(\hat{\tau}(i) + \mathcal{I})$. Hence, we get that $\sup R - \hat{\tau}(\max N) \succ \sup R - \inf(\hat{\tau}(i) + \mathcal{I}) \geq \sup(\hat{\tau}(i) + \mathcal{I}) - \inf(\hat{\tau}(i) + \mathcal{I}) \geq \sup \mathcal{I} - \inf \mathcal{I} > \Delta\hat{\tau}$, which is a contradiction by Assumption 5.3.1.
2. There exists some sample $j \geq i$ such that the time interval $\hat{\tau}(i) + \mathcal{I}$ fits between the samples j and $j + 1$. Formally, there exists $j \in N_{\geq i}$ such that $\hat{\tau}(j) \prec \inf(\hat{\tau}(i) + \mathcal{I})$ and $\sup(\hat{\tau}(i) + \mathcal{I}) \prec \hat{\tau}(j + 1)$, where $\prec \in \{<, \leq\}$ depending on the constraints of \mathcal{I} . That is, $\hat{\tau}(j + 1) - \hat{\tau}(j) \succ \sup(\hat{\tau}(i) + \mathcal{I}) - \inf(\hat{\tau}(i) + \mathcal{I}) = \sup \mathcal{I} - \inf \mathcal{I} > \Delta\hat{\tau}$, which is a contradiction by definition (equation (5.2)).

Note that the case where all the samples $j \geq i$ map to points in time that happen later than the maximum required time by $\hat{\tau}(i) + \mathcal{I}$ cannot be considered since the time $\hat{\tau}(i)$ cannot occur after the time interval $\hat{\tau}(i) + \mathcal{I}$. Thus, $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}) \neq \emptyset$.

9.1.3 Proof of Lemma 5.3.2

If R is unbounded, then the result is immediate from Lemma 5.3.1. If now R is bounded, we have by definition that $\sum_{j \geq k} \sup \mathcal{I}_j \leq \mathbf{dur}(\mathbf{str}_{\Delta\hat{\tau}}(\phi)) < \sup R$. Thus, for any $i \in I$, we have $\hat{\tau}(i) + \sup \mathcal{I}_k < \sup R$ and, therefore, $(\hat{\tau}(i) + \mathcal{I}_k) \subseteq R$. Thus, $(\hat{\tau}(i) + \mathcal{I}_k) = (\hat{\tau}(i) +_R \mathcal{I}_k)$. The result follows by Lemma 5.3.1. Since $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$, we also get that $\hat{\tau}^{-1}([0, \sum_{j \geq k} \sup \mathcal{I}_j]) \neq \emptyset$ (note that by assumption $\hat{\tau}^{-1}(T) \neq \emptyset$).

9.1.4 Proof of Theorem 5.3.1

The proof of the theorem is by induction on the structure of formula ϕ . In the following, we set $\sigma = s \circ \hat{\tau}$, $\mu = (\sigma, \hat{\tau})$ and $T_i = [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R$ for $i \in N$.

Case $\phi = \top$: $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\top) \rrbracket_D(\mu, i) = +\infty > \mathcal{E}(\Delta\hat{\tau})$. Therefore, for all $t \in T_i$, we have $\langle\langle \top \rangle\rangle_C(s, t) = \top$.

Case $\phi = p \in AP$: $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(p) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$, i.e., $\mathbf{depth}_d(\sigma(i), \mathcal{O}(p)) > \mathcal{E}(\Delta\hat{\tau})$. Therefore, $d(\sigma(i), x) > \mathcal{E}(\Delta\hat{\tau})$ for any $x \in \overline{X \setminus \mathcal{O}(p)}$. Moreover by Assumption 5.2.1, we get that $d(\sigma(i), s(t)) \leq \mathcal{E}(\Delta\hat{\tau})$ for all $t \in T_i$ and $d(\sigma(i), s(t)) \leq \mathcal{E}(\Delta\hat{\tau}) < d(\sigma(i), x)$. Also, since d is a metric : $d(\sigma(i), x) \leq d(\sigma(i), s(t)) + d(s(t), x)$. Hence, $d(s(t), x) > 0$. Since this holds for any $x \in \overline{X \setminus \mathcal{O}(p)}$, we conclude that $s(t) \in \mathcal{O}(p)$ and, thus, $\langle\langle p \rangle\rangle_C(s, t) = \top$ for all $t \in T_i$.

Case $\phi = \neg p \in AP$: $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\neg p) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$, i.e., $\mathbf{dist}_d(\sigma(i), \mathcal{O}(p)) > \mathcal{E}(\Delta\hat{\tau})$. The proof is similar to the previous case.

Case $\phi = \phi_1 \wedge \phi_2$: We have that $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \wedge \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$. Thus, both $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$ and $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$. By the induction hypothesis, we get that for all $t \in T_i$, we have $\langle\langle \phi_1 \rangle\rangle_C(s, t) = \top$ and for all $t \in T_i$, we have $\langle\langle \phi_2 \rangle\rangle_C(s, t) = \top$. That is, for all $t \in T_i$, we have $\langle\langle \phi_1 \rangle\rangle_C(s, t) = \top$ and $\langle\langle \phi_2 \rangle\rangle_C(s, t) = \top$. Hence, for all $t \in T_i$, we have $\langle\langle \phi \rangle\rangle_C(s, t) = \top$.

Case $\phi = \phi_1 \vee \phi_2$: The proof is similar to the previous case.

Case $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: We know that $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{U}_{C_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$. By Lemma 5.3.2, we have $J = \hat{\tau}^{-1}(\hat{\tau}(i) + C_{d_1}(\mathcal{I}, \Delta\hat{\tau})) \neq \emptyset$. By the definition of until, there exists some $j \in J$ such that $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, j) > \mathcal{E}(\Delta\hat{\tau})$ and for all k such that $i \leq k < j$, we have $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \rrbracket_D(\mu, k) > \mathcal{E}(\Delta\hat{\tau})$. By the induction hypothesis, we get that $\langle\langle \phi_2 \rangle\rangle_C(s, t) = \top$ for all $t \in T_j$ and $\langle\langle \phi_1 \rangle\rangle_C(s, t) = \top$ for all $t \in T_k$ and for all $k \in [i, j)$. We set $t' = \hat{\tau}(j)$. Note that for all $t \in T_i$, we have $\hat{\tau}(j) \in \hat{\tau}(i) + C_{d_1}(\mathcal{I}, \Delta\hat{\tau}) \subseteq (t + \mathcal{I})$. But $\hat{\tau}(j) \in \hat{\tau}(i) +_R C_{d_1}(\mathcal{I}, \Delta\hat{\tau})$, thus we have $t' = \hat{\tau}(j) \in (t +_R \mathcal{I}) \neq \emptyset$. Also, since $\hat{\tau}(j) \leq \hat{\tau}(j-1) + \Delta\hat{\tau}$, we get that for all $t'' \in (t, t')$, we have $\langle\langle \phi_1 \rangle\rangle_C(s, t'') = \top$. Hence, we conclude that $\langle\langle \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rangle\rangle_C(s, t) = \top$ for all $t \in T_i$.

Case $\phi = \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2$: We have $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \mathcal{R}_{E_{d_1}(\mathcal{I}, \Delta\hat{\tau})} \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, i) > \mathcal{E}(\Delta\hat{\tau})$. By Lemma 5.3.2, we have $J = \hat{\tau}^{-1}(\hat{\tau}(i) + E_{d_1}(\mathcal{I}, \Delta\hat{\tau})) \neq \emptyset$. By the definition of release, for all $j \in J$, we have $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_2) \rrbracket_D(\mu, j) > \mathcal{E}(\Delta\hat{\tau})$ or there exists k such that $i \leq k < j$ and $\llbracket \mathbf{str}_{\Delta\hat{\tau}}(\phi_1) \rrbracket_D(\mu, k) > \mathcal{E}(\Delta\hat{\tau})$. By the induction hypothesis, we get that for all $j \in J$, we have $\langle\langle \phi_2 \rangle\rangle_C(s, t) = \top$ for all $t \in T_j$ and $\langle\langle \phi_1 \rangle\rangle_C(s, t) = \top$ for all $t \in T_k$. Let $j_m = \min J$ and $j_M = \max J$. For all $t' \in [\hat{\tau}(j_m) - \Delta\hat{\tau}, \hat{\tau}(j_M) + \Delta\hat{\tau}] \cap R$, we have $\langle\langle \phi_2 \rangle\rangle_C(s, t') = \top$. Moreover, for all $t \in T_i$, we have $(t + \mathcal{I}) \subseteq \hat{\tau}(i) + E_{d_1}(\mathcal{I}, \Delta\hat{\tau})$. But by Lemma 5.3.2, we get $\hat{\tau}(i) +_R E_{d_1}(\mathcal{I}, \Delta\hat{\tau}) = \hat{\tau}(i) + E_{d_1}(\mathcal{I}, \Delta\hat{\tau})$. We conclude that $(t +_R \mathcal{I}) \neq \emptyset$ since $(t +_R \mathcal{I}) \subseteq \hat{\tau}(i) +_R E_{d_1}(\mathcal{I}, \Delta\hat{\tau})$. Hence, for all $t \in T_i$, for all $t' \in (t +_R \mathcal{I})$, we have $\langle\langle \phi_2 \rangle\rangle_C(s, t') = \top$ or there exists some $t'' \in (t, t')$ such that $\langle\langle \phi_1 \rangle\rangle_C(s, t'') = \top$. Hence, $\langle\langle \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2 \rangle\rangle_C(s, t) = \top$ for all $t \in T_i$.

9.1.5 Proof of Lemma 5.4.2

If R is unbounded, then the result is immediate from Lemma 5.4.1. If R is bounded, we need to consider two cases.

- Case 1 of Assumption 5.4.3: Consider any $t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R$. We have that $t + \sup \mathcal{I}_k \leq \hat{\tau}(i) + \Delta\hat{\tau} + \sup \mathcal{I}_k \leq \mathbf{dur}(\phi) + \Delta\hat{\tau} < \sup R$. Thus, $t + {}_R\mathcal{I}_k \neq \emptyset$ and $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$.
- Case 2 of Assumption 5.4.3: Since $0 \in \mathcal{I}$, we immediately get that $i \in \hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$ and that for all $t \in [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R$, $t \in (t + {}_R\mathcal{I}_k) \neq \emptyset$.

Since $\hat{\tau}^{-1}(T) \neq \emptyset$ and $\hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I}_k) \neq \emptyset$, we also get that $\hat{\tau}^{-1}([0, \sum_{j \geq k} \sup \mathcal{I}_j]) \neq \emptyset$.

9.1.6 Proof of Theorem 5.4.1

The proof is by induction on the structure of formula ϕ . In the following, we always set $\sigma = s \circ \hat{\tau}$, $\mu = (\sigma, \hat{\tau})$ and $\psi = \mathbf{mtc}(\phi)$. For the sake of brevity, we also define

$$T_i = [\hat{\tau}(i) - \Delta\hat{\tau}, \hat{\tau}(i) + \Delta\hat{\tau}] \cap R$$

for $i \in N$. By Assumption 5.4.2, there exists some $\alpha \in \mathbb{Q}_{>0}$ such that $\hat{\tau}(i) = ai$ for $i \in N$. Thus, $\Delta\hat{\tau} = \alpha$ and

$$T_i = [a(i-1), a(i+1)] \cap R$$

Case $\phi = \top$: We have $\psi = \top$. By definition, for all $t \in T_i$, we have $\llbracket \top \rrbracket_C(s, t) = +\infty$ and, also, $\llbracket \top \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) = \llbracket \top \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) = +\infty$.

Case $\phi = p \in AP$: We have $\psi = p$. In the following, we let $t \in T_i$. By Assumption 5.2.1, we have

$$d(\sigma(i), s(t)) \leq \mathcal{E}(\Delta\hat{\tau}) \tag{9.1}$$

We must consider 4 cases according to the values of $\llbracket p \rrbracket_D(\mu, i)$ and $\llbracket p \rrbracket_C(s, t)$.

1. Assume that $s(t), \sigma(i) \in \mathcal{O}(p)$, that is, $\llbracket p \rrbracket_D(\mu, i) = \mathbf{dist}_d(\sigma(i), X \setminus \mathcal{O}(p))$ and

$\llbracket p \rrbracket_C(s, t) = \mathbf{dist}_d(s(t), X \setminus \mathcal{O}(p))$. Since $\mathbf{dist}_d(\sigma(i), X \setminus \mathcal{O}(p)) \leq d(\sigma(i), x)$ for any $x \in \overline{X \setminus \mathcal{O}(p)}$, from the triangle inequality, we have

$$\begin{aligned} \mathbf{dist}_d(\sigma(i), X \setminus \mathcal{O}(p)) &\leq d(\sigma(i), x) \leq d(\sigma(i), s(t)) + d(s(t), x) \stackrel{(9.1)}{\implies} \\ \llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) &\leq d(s(t), x) \end{aligned}$$

That is, $\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau})$ is a lower bound on $d(s(t), x)$ over the set $\overline{X \setminus \mathcal{O}(p)}$ and, thus, $\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau})$ is less than or equal to the greatest lower bound (glb) on $d(s(t), x)$ over the set $\overline{X \setminus \mathcal{O}(p)}$ or

$$\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) \leq \inf\{d(s(t), x) \mid x \in \overline{X \setminus \mathcal{O}(p)}\} = \llbracket p \rrbracket_C(s, t)$$

By symmetry, we get

$$\llbracket p \rrbracket_C(s, t) - \mathcal{E}(\Delta \hat{\tau}) \leq \llbracket p \rrbracket_D(\mu, i) \implies \llbracket p \rrbracket_C(s, t) \leq \llbracket p \rrbracket_D(\mu, i) + \mathcal{E}(\Delta \hat{\tau})$$

2. Assume that $s(t), \sigma(i) \in X \setminus \mathcal{O}(p)$, i.e., $\llbracket p \rrbracket_D(\mu, i) = -\mathbf{dist}_d(\sigma(i), \mathcal{O}(p))$ and $\llbracket p \rrbracket_C(s, t) = -\mathbf{dist}_d(s(t), \mathcal{O}(p))$. Since $\mathbf{dist}_d(\sigma(i), \mathcal{O}(p)) \leq d(\sigma(i), x)$ for any $x \in \overline{\mathcal{O}(p)}$, using the triangle inequality and the glb argument from the previous case, we have

$$\begin{aligned} \mathbf{dist}_d(\sigma(i), \mathcal{O}(p)) &\leq d(\sigma(i), x) \leq d(\sigma(i), s(t)) + d(s(t), x) \stackrel{(9.1)}{\implies} \\ -\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) &\leq d(s(t), x) \stackrel{(glb)}{\implies} \\ -\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) &\leq \mathbf{dist}_d(s(t), \mathcal{O}(p)) = -\llbracket p \rrbracket_C(s, t) \implies \\ \llbracket p \rrbracket_C(s, t) &\leq \llbracket p \rrbracket_D(\mu, i) + \mathcal{E}(\Delta \hat{\tau}) \end{aligned}$$

By symmetry, we get

$$\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) \leq \llbracket p \rrbracket_C(s, t)$$

3. Now, we prove the case where $\sigma(i) \in \mathcal{O}(p)$ and $s(t) \in X \setminus \mathcal{O}(p)$. Let $\varepsilon_D = \llbracket p \rrbracket_D(\mu, i)$ and $\varepsilon_C = -\llbracket p \rrbracket_C(s, t)$.

- Case $\varepsilon_D > 0$ and $\varepsilon_C > 0$: let $B_D = B_d(\sigma(i), \varepsilon_D)$ and $B_C = B_d(s(t), \varepsilon_C)$. Since $\sigma(i) \in \mathcal{O}(p)$ and $s(t) \in X \setminus \mathcal{O}(p)$, we have $B_D \subseteq \mathcal{O}(p)$ and $B_C \subseteq X \setminus \mathcal{O}(p)$. Hence, $B_D \cap B_C = \emptyset$, which implies that $\varepsilon_D + \varepsilon_C \leq d(\sigma(i), s(t))$.
- Case $\varepsilon_D = 0$ and $\varepsilon_C > 0$: $\sigma(i)$ is on the boundary of the set $\mathcal{O}(p)$ and, thus, on the boundary of the set $X \setminus \mathcal{O}(p)$. Since ε_C is the shortest distance from $s(t)$ to the boundary of the set $X \setminus \mathcal{O}(p)$, we get that $d(\sigma(i), s(t)) \geq \varepsilon_C = \varepsilon_D + \varepsilon_C$.
- Case $\varepsilon_D > 0$ and $\varepsilon_C = 0$: similarly to the previous case, we have $d(\sigma(i), s(t)) \geq \varepsilon_D = \varepsilon_D + \varepsilon_C$.
- Case $\varepsilon_D = 0$ and $\varepsilon_C = 0$: this case is included in the cases (1) or (2) above since both points belong to the same sets.

Therefore in every case, by using the inequality (9.1), we get

$$\varepsilon_D + \varepsilon_C \leq \mathcal{E}(\Delta \hat{\tau}) \implies \llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) \leq \llbracket p \rrbracket_C(s, t)$$

Moreover, since $\llbracket p \rrbracket_D(\mu, i) \geq 0$ and $\llbracket p \rrbracket_C(s, t) \leq 0$, we immediately get

$$\llbracket p \rrbracket_C(s, t) \leq \llbracket p \rrbracket_D(\mu, i) + \mathcal{E}(\Delta \hat{\tau})$$

4. Similarly to the previous case, when $s(t) \in \mathcal{O}(p)$ and $\sigma(i) \in X \setminus \mathcal{O}(p)$, then

$$\varepsilon_D = -\llbracket p \rrbracket_D(\mu, i) \text{ and } \varepsilon_C = \llbracket p \rrbracket_C(s, t)$$

$$\varepsilon_D + \varepsilon_C \leq \mathcal{E}(\Delta\hat{\tau}) \implies \llbracket p \rrbracket_C(s, t) \leq \llbracket p \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})$$

Moreover, since $\llbracket p \rrbracket_D(\mu, i) \leq 0$ and $\llbracket p \rrbracket_C(s, t) \geq 0$, we immediately get

$$\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket p \rrbracket_C(s, t)$$

Therefore, we conclude that for all $t \in T_i$ we have

$$\llbracket p \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket p \rrbracket_C(s, t) \leq \llbracket p \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})$$

Case $\phi = \neg\phi_1$: Let $\psi_1 = \mathbf{mtc}(\phi_1)$. By the induction hypothesis, for all $t \in T_i$, we have

$$\begin{aligned} & \llbracket \psi_1 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_1 \rrbracket_C(s, t) \leq \llbracket \psi_1 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) \implies \\ & -\llbracket \neg\psi_1 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq -\llbracket \neg\phi_1 \rrbracket_C(s, t) \leq -\llbracket \neg\psi_1 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) \implies \\ & \llbracket \neg\psi_1 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) \geq \llbracket \neg\phi_1 \rrbracket_C(s, t) \geq \llbracket \neg\psi_1 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \implies \\ & \llbracket \psi \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi \rrbracket_C(s, t) \leq \llbracket \psi \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau}) \end{aligned}$$

Case $\phi = \phi_1 \vee \phi_2$: Let $\psi_1 = \mathbf{mtc}(\phi_1)$ and $\psi_2 = \mathbf{mtc}(\phi_2)$. By the induction hypothesis, we get that for $j = 1, 2$, for all $t \in T_i$, we have

$$\llbracket \psi_j \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_j \rrbracket_C(s, t) \leq \llbracket \psi_j \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})$$

Since \sqcup is monotonic with respect to the relation \leq , for all $t \in T_i$, we have

$$\begin{aligned}
& (\llbracket \psi_1 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau})) \sqcup (\llbracket \psi_2 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau})) \leq \llbracket \phi_1 \rrbracket_C(s, t) \sqcup \llbracket \phi_2 \rrbracket_C(s, t) \leq \\
& \leq (\llbracket \psi_1 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})) \sqcup (\llbracket \psi_2 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})) \implies \\
& (\llbracket \psi_1 \rrbracket_D(\mu, i) \sqcup \llbracket \psi_2 \rrbracket_D(\mu, i)) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_1 \rrbracket_C(s, t) \sqcup \llbracket \phi_2 \rrbracket_C(s, t) \leq \\
& \leq (\llbracket \psi_1 \rrbracket_D(\mu, i) \sqcup \llbracket \psi_2 \rrbracket_D(\mu, i)) + \mathcal{E}(\Delta\hat{\tau}) \implies \\
& \llbracket \psi_1 \vee \psi_2 \rrbracket_D(\mu, i) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_1 \vee \phi_2 \rrbracket_C(s, t) \leq \llbracket \psi_1 \vee \psi_2 \rrbracket_D(\mu, i) + \mathcal{E}(\Delta\hat{\tau})
\end{aligned}$$

Case $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$: We have $\psi = \psi_1 \bar{\mathcal{U}}_{\mathcal{I}} \psi_2$, where $\psi_1 = \mathbf{mtc}(\phi_1)$ and $\psi_2 = \mathbf{mtc}(\phi_2)$. Formally, the discrete-time robust semantics of the temporal operator $\bar{\mathcal{U}}$ are defined as

$$\llbracket \phi_1 \bar{\mathcal{U}}_{\mathcal{I}} \phi_2 \rrbracket_D(\mu, i) := \bigsqcup_{j \in \tau^{-1}(\tau(i) + \mathcal{I})} (\llbracket \phi_2 \rrbracket_D(\mu, j) \sqcap \prod_{i \leq k \leq j} \llbracket \phi_1 \rrbracket_D(\mu, k))$$

Let $J = \hat{\tau}^{-1}(\hat{\tau}(i) + \mathcal{I})$. By Lemma 5.3.2, we know that the set J is nonempty. Now let $t \in T_i$ and consider any $t' \in (t +_R \mathcal{I})$, which is a non-empty set by Lemma 5.3.2. Since $t +_R \mathcal{I} \subseteq \cup_{j \in J} T_j$, there exists some $j \in J$ such that $t' \in T_j$. Note that for all $l = i, i + 1, \dots, \max J$, we have $T_j \neq \emptyset$. By the induction hypothesis, for all $\tilde{t} \in T_j$, we get that

$$\llbracket \psi_2 \rrbracket_D(\mu, j) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_2 \rrbracket_C(s, \tilde{t}) \leq \llbracket \psi_2 \rrbracket_D(\mu, j) + \mathcal{E}(\Delta\hat{\tau}) \quad (9.2)$$

and for all $k \in [i, j]$, for all $\bar{t} \in T_k$, we have

$$\llbracket \psi_1 \rrbracket_D(\mu, k) - \mathcal{E}(\Delta\hat{\tau}) \leq \llbracket \phi_1 \rrbracket_C(s, \bar{t}) \leq \llbracket \psi_1 \rrbracket_D(\mu, k) + \mathcal{E}(\Delta\hat{\tau}) \quad (9.3)$$

Let $Q_k^{t,t'} = T_k \cap (t, t')$. Note that for any $k \in [i, j]$, we have $Q_k^{t,t'} \neq \emptyset$. From (9.3), for any $k \in [i, j]$, for all $\bar{t} \in Q_k^{t,t'}$, we have

$$\prod_{t'' \in Q_k^{t,t'}} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \llbracket \phi_1 \rrbracket_C(s, \bar{t}) \leq \llbracket \psi_1 \rrbracket_D(\mu, k) + \mathcal{E}(\Delta \hat{\tau}) \quad (9.4)$$

Also, since $\llbracket \psi_1 \rrbracket_D(\mu, k) - \mathcal{E}(\Delta \hat{\tau})$ is a lower bound on $\llbracket \phi_1 \rrbracket_C(s, \cdot)$ over the set $Q_k^{t,t'}$ and $\prod_{t'' \in Q_k^{t,t'}} \llbracket \phi_1 \rrbracket_C(s, t'')$ is the greatest lower bound, we have

$$\llbracket \psi_1 \rrbracket_D(\mu, k) - \mathcal{E}(\Delta \hat{\tau}) \leq \prod_{t'' \in Q_k^{t,t'}} \llbracket \phi_1 \rrbracket_C(s, t'') \quad (9.5)$$

Then, using the last two inequalities and the monotonicity of \prod with respect to the ordering relation \leq , we get

$$\begin{aligned} \prod_{k \in [i, j]} (\llbracket \psi_1 \rrbracket_D(\mu, k) - \mathcal{E}(\Delta \hat{\tau})) &\leq \prod_{k \in [i, j]} \prod_{t'' \in Q_k^{t,t'}} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \\ &\leq \prod_{k \in [i, j]} (\llbracket \psi_1 \rrbracket_D(\mu, k) + \mathcal{E}(\Delta \hat{\tau})) \end{aligned}$$

Note that $\cup_{k=i}^j Q_k^{t,t'} = (t, t')$. We should point out that this is true only because we are using the matching until operator. If instead we were using the non-matching operator, then there would exist some $t \in T_i$ and some $t' \in (t +_R \mathcal{I})$ such that $\cup_{k=i}^{j-1} Q_k^{t,t'} \subset (t, t')$. Thus, we have

$$\begin{aligned} \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) - \mathcal{E}(\Delta \hat{\tau}) &\leq \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \\ &\leq \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) + \mathcal{E}(\Delta \hat{\tau}) \end{aligned} \quad (9.6)$$

Again, by using the monotonicity of \sqcap and by pulling out the constant $\mathcal{E}(\Delta\hat{\tau})$ from the min operator, from (9.2) and (9.6), for any $t' \in T_j$, we have

$$\begin{aligned} & \left(\llbracket \psi_2 \rrbracket_D(\mu, j) \sqcap \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) \right) - \mathcal{E}(\Delta\hat{\tau}) \leq \\ & \leq \llbracket \phi_2 \rrbracket_C(s, t') \sqcap \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \\ & \leq \left(\llbracket \psi_2 \rrbracket_D(\mu, j) \sqcap \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) \right) + \mathcal{E}(\Delta\hat{\tau}) \end{aligned}$$

Let $P_j^t = T_j \cap (t +_R \mathcal{I})$. Note that if Assumptions 5.4.1 and 5.4.2 do not hold, then it is not true that $P_j^t \neq \emptyset$. Next, we prove by contradiction that $P_j^t \neq \emptyset$ since Assumptions 5.4.1 to 5.4.3 hold.

Claim 9.1.1. *For any $j \in J$, the set $P_j^t = T_j \cap (t +_R \mathcal{I})$ is not empty.*

Proof. First note that since $t \in T_i$, we have

$$\max\{0, \alpha(i-1)\} \leq t \leq \min\{\alpha(i+1), \sup R\} \quad (9.7)$$

Moreover, we have $T_j \neq \emptyset$ and $(t +_R \mathcal{I}) \neq \emptyset$. Assume now that $P_j^t = \emptyset$. We consider two cases which depend on \mathcal{I} :

1. $\mathcal{I} = [\alpha i_1, +\infty)$ for some $i_1 \in \mathbb{N}$: This is possible only if R is unbounded or $i_1 = 0$, i.e., $\mathcal{I} = [0, +\infty)$. Since $j \in J$, we have

$$\begin{aligned} \hat{\tau}(j) \in (\hat{\tau}(i) +_R \mathcal{I}) & \implies \alpha j \in (\alpha i +_R [\alpha i_1, +\infty)) \implies \\ \alpha j \in [\alpha i + \alpha i_1, +\infty) \cap R & \implies \alpha j \geq \alpha(i + i_1) \implies i + i_1 \leq j \end{aligned} \quad (9.8)$$

Also, $P_j^t = \emptyset$ implies that

- either $\sup T_j < \inf(t +_R \mathcal{I})$, that is,

$$0 \leq \min\{\alpha(j+1), \sup R\} < \inf([t + \alpha i_1, +\infty) \cap R) = \min\{0, t + \alpha i_1\} \implies \\ \min\{\alpha(j+1), \sup R\} < t + \alpha i_1$$

$\sup R < t + \alpha i_1$ is a contradiction, because in this case $t +_R \mathcal{I} = \emptyset$. Thus,

$$\alpha(j+1) < t + \alpha i_1 \xrightarrow{(9.8)} \alpha(i + i_1 + 1) < t + \alpha i_1 \implies \alpha(i+1) < t$$

which is a contradiction by eq. (9.7).

- or $\sup(t +_R \mathcal{I}) < \inf T_j$. If $\sup R = +\infty$, then this is immediately a contradiction. If R is bounded, then $\sup(t +_R [0, +\infty)) = \sup R < \inf T_j$, which is a contradiction since $T_j \neq \emptyset$.

2. $\mathcal{I} = [\alpha i_1, \alpha i_2]$ for some $i_1, i_2 \in \mathbb{N}$ such that $i_1 < i_2$: R can be bounded or unbounded. In either case, we have $t +_R \mathcal{I} \subseteq R$ by assumption and, thus, $t +_R \mathcal{I} = t + \mathcal{I}$. Since $j \in J$, we have

$$\hat{\tau}(j) \in (\hat{\tau}(i) + \mathcal{I}) \implies \alpha j \in (\alpha i + [\alpha i_1, \alpha i_2]) \implies \\ \alpha j \in [\alpha i + \alpha i_1, \alpha i + \alpha i_2] \implies \\ \alpha(i + i_1) \leq \alpha j \leq \alpha(i + i_2) \implies i + i_1 \leq j \leq i + i_2 \quad (9.9)$$

Also, $P_j^t = \emptyset$ implies that $\sup T_j < \inf(t + \mathcal{I})$ or $\sup(t + \mathcal{I}) < \inf T_j$. The case $\sup T_j < \inf(t + \mathcal{I})$ is the same as above. For the case $\sup(t + \mathcal{I}) < \inf T_j$, we have

$$\sup[t + \alpha i_1, t + \alpha i_2] < \alpha(j-1) \implies t + \alpha i_2 < \alpha(j-1) \xrightarrow{(9.9)}$$

$$t + \alpha i_2 < \alpha(i + i_2 - 1) \implies t < \alpha(i - 1)$$

which is a contradiction by eq. (9.7). \square

Since $\cup_{j \in J} P_j^t = (t + {}_R\mathcal{I})$, similarly to the derivation of (9.6), for any $t \in T_i$, we get

$$\begin{aligned} & \left(\bigsqcup_{j \in J} \llbracket \psi_2 \rrbracket_D(\mu, j) \sqcap \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) \right) - \mathcal{E}(\Delta \hat{\tau}) \leq \\ & \leq \bigsqcup_{t' \in (t + {}_R\mathcal{I})} \llbracket \phi_2 \rrbracket_C(s, t') \sqcap \prod_{t'' \in (t, t')} \llbracket \phi_1 \rrbracket_C(s, t'') \leq \\ & \leq \left(\bigsqcup_{j \in J} \llbracket \psi_2 \rrbracket_D(\mu, j) \sqcap \prod_{k \in [i, j]} \llbracket \psi_1 \rrbracket_D(\mu, k) \right) + \mathcal{E}(\Delta \hat{\tau}) \implies \\ & \llbracket \psi \rrbracket_D(\mu, i) - \mathcal{E}(\Delta \hat{\tau}) \leq \llbracket \phi \rrbracket_C(s, t) \leq \llbracket \psi \rrbracket_D(\mu, i) + \mathcal{E}(\Delta \hat{\tau}) \end{aligned}$$

9.2 Proofs of Chapter 6

9.2.1 Proof of Theorem 6.2.1

Equation (6.3) is equivalent to eq. (4.8). The fact that eq. (6.4) implies equation

$$\forall \hat{p}_1 \in P_1 \cdot \forall \hat{p}_2 \in P_2 \cdot A_\star^T(\hat{p}_1, \hat{p}_2)M + MA_\star(\hat{p}_1, \hat{p}_2) \leq 0,$$

which is equivalent to eq. (4.9), is established in Theorem 1 in [102].

9.2.2 Proof of Proposition 6.2.1

First, let us rewrite the function \mathcal{V} as

$$\mathcal{V}(x_i, x_u) = \begin{bmatrix} x_i^T & x_u^T \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} x_i \\ x_u \end{bmatrix} = x_i^T M_{11} x_i + 2x_i^T M_{12} x_u + x_u^T M_{22} x_u$$

Note that \mathcal{V} is convex in both arguments since X_0 is convex and M_{11} and M_{22} are positive semidefinite (see §3.1.4 in [26]) as principal minors of the positive semidefinite matrix M (see §7.1.2 in [103]). Let us first consider the game

$$\max_{x_u^0 \in \mathcal{EP}(X_0)} \inf_{x_i^0 \in X_0} \mathcal{V}(x_i^0, x_u^0).$$

The function

$$h(x) = \inf_{x_i^0 \in X_0} \mathcal{V}(x_i^0, x)$$

is a convex function since \mathcal{V} is convex in both of its arguments and X_0 is a convex and nonempty set (see §3.2.5 in [26]). Note that h attains its minimum in the set X_0 . The supremum of a convex function over a (bounded) polyhedral set is achieved on one of its extreme points (see Theorem 32.2 in [166]). Thus, we have

$$\sup_{x_u^0 \in X_0} h(x_u^0) = \max_{x_u^0 \in \mathcal{EP}(X_0)} h(x_u^0).$$

Since square root is a strictly increasing function, we have

$$\beta(x) = \arg \min_{x_i^0 \in X_0} \mathcal{F}(x_i^0, x) = \arg \min_{x_i^0 \in X_0} \mathcal{V}(x_i^0, x)$$

for any $x \in X_0$ (see §4.1.3 in [26]). Now, set

$$x_u^* = \arg \max_{x_u^0 \in X_0} \mathcal{V}(\beta(x_u^0), x_u^0).$$

Thus, for all $x_u \in X_0$, we have $\mathcal{V}(\beta(x_u^*), x_u^*) \geq \mathcal{V}(\beta(x_u), x_u)$ and because the square root function is strictly increasing, we get $\mathcal{F}(\beta(x_u^*), x_u^*) \geq \mathcal{F}(\beta(x_u), x_u)$ for all $x_u \in X_0$. In other words, $\mathcal{F}(\beta(x_u^*), x_u^*) \geq \sup_{x_u^0 \in X_0} \mathcal{F}(\beta(x_u^0), x_u^0)$. But also by definition $\mathcal{F}(\beta(x_u^*), x_u^*) \leq \sup_{x_u^0 \in X_0} \mathcal{F}(\beta(x_u^0), x_u^0)$. Therefore,

$$\mathcal{F}(\beta(x_u^*), x_u^*) = \sup_{x_u^0 \in X_0} \inf_{x_i^0 \in X_0} \mathcal{F}(x_i^0, x_u^0),$$

that is, the optimal points x_u^* and $x_i^* = \beta(x_u^*)$ solve also the game

$$\sup_{x_u^0 \in X_0} \inf_{x_i^0 \in X_0} \mathcal{F}(x_i^0, x_u^0).$$

Similarly, we get the optimal points for the game

$$\max_{x_i^0 \in \mathcal{EP}(X_0)} \inf_{x_u^0 \in X_0} \mathcal{V}(x_i^0, x_u^0)$$

and, thus, by solving the games in (6.7) we can get the upper bound δ for (4.3).

9.2.3 Proof of Theorem 6.3.1

From equation (4.7), it follows that for all $t \in \mathbb{R}_{\geq 0}$,

$$\frac{d\mathcal{F}(x_1(t), x_2(t))}{dt} = \frac{\partial \mathcal{F}}{\partial x_1}(x_1(t), x_2(t)) \cdot f(x_1(t)) + \frac{\partial \mathcal{F}}{\partial x_2}(x_1(t), x_2(t)) \cdot f(x_2(t)) \leq 0,$$

that is, the value of \mathcal{F} is non-increasing with the parallel evolution of the two state trajectories. Then, from equation (4.6), for all $t \in \mathbb{R}_{\geq 0}$,

$$\|y_1(t) - y_2(t)\| = \|Cx_1(t) - Cx_2(t)\| \leq \mathcal{F}(x_1(t), x_2(t)) \leq \mathcal{F}(x_1(0), x_2(0)).$$

Therefore,

$$\begin{aligned} \forall t \in R. \|y_1(t) - y_2(t)\| \leq \mathcal{F}(x_1(0), x_2(0)) &\implies \rho(y_1, y_2) \leq \mathcal{F}(x_1(0), x_2(0)) \implies \\ \rho(y_1, y_2) < \llbracket \phi, \mathcal{O} \rrbracket_C(y_1) - \delta &\implies \rho(y_1, y_2) + \delta < \varepsilon \end{aligned}$$

where $\varepsilon = \llbracket \phi, \mathcal{O} \rrbracket_C(y_1)$. Now, consider any observation trajectory y in the δ neighborhood of y_2 , i.e, $y \in B_\rho(y_2, \delta)$. We have

$$\rho(y_1, y) \leq \rho(y_1, y_2) + \rho(y_2, y) < \rho(y_1, y_2) + \delta < \varepsilon$$

Thus, $y \in B_\rho(y_1, \varepsilon)$. But by the definition of the robustness estimate, we get $B_\rho(y_1, \varepsilon) \subseteq \mathcal{L}(\phi, \mathcal{O})$, which implies $y \in \mathcal{L}(\phi, \mathcal{O})$ and, thus, $B_\rho(y_2, \delta) \subseteq \mathcal{L}(\phi, \mathcal{O})$. From the last set inclusion and the definition of signed distance we get the desired result, that is, $\mathbf{Dist}_\rho(y_2, \mathcal{L}(\phi, \mathcal{O})) \geq \delta$.

9.2.4 Proof of Proposition 6.3.1

\mathcal{F} is continuous on $X^0 \times X^0$ which is compact, therefore \mathcal{F} is uniformly continuous on $X^0 \times X^0$. Hence, for all ε , there exists ν such that

$$\forall x, x', z, z' \in X^0, \|x - x'\| \leq \nu \text{ and } \|z - z'\| \leq \nu \implies |\mathcal{F}(x, z) - \mathcal{F}(x', z')| \leq \varepsilon.$$

Particularly, by setting $x' = z = z'$ and remarking that $\mathcal{F}(x', x') = 0$, we have

$$\forall x, x' \in X^0, \|x - x'\| \leq \nu \implies \mathcal{F}(x, x') \leq \varepsilon.$$

Now, let us assume that for all finite set of points $\{x_1, \dots, x_k\} \subseteq X^0$, there exists $x_{r+1} \in X^0$, such that for all x_i , $\|x - x_i\| \geq \nu$. Then, starting from a point $x_1 \in X^0$, we can construct a sequence $\{x_i\}_{i \in \mathbb{N}}$ such that for all $i, j \in \mathbb{N}$, $i \neq j$, we have $\|x_i - x_j\| \geq \nu$. Therefore, we cannot extract a converging subsequence of $\{x_i\}_{i \in \mathbb{N}}$ and X^0 cannot be compact. Hence, we have proved by contradiction that there exists a finite set of points $\{x_1, \dots, x_k\} \subseteq X^0$ such that for all $x \in X^0$, there exists x_i , such that $\|x - x_i\| \leq \nu$ which allows us to conclude (6.8).

9.2.5 Proof of Theorem 6.3.2

Let $x \in \Lambda(\overline{\Sigma}^{\text{LTI}})$ be a state trajectory and $y \in \mathcal{L}(\overline{\Sigma}^{\text{LTI}})$ be the associated observation trajectory such that $y(t) = Cx(t)$. From Proposition 6.3.1, there exists x_i such that $\mathcal{F}(x(0), x_i(0)) \leq \varepsilon$. Then, from Theorem 6.3.1 follows that $\mathbf{Dist}_\rho(y, \mathcal{L}(\phi, \mathcal{O})) \geq \delta$. Since this holds for any $x \in \Lambda(\overline{\Sigma}^{\text{LTI}})$, the desired results follows.

9.3 Proofs of Chapter 7

9.3.1 Proof of Proposition 7.2.1

From Definition 7.2.2, we have that for all $t \geq 0$, $(z(t), \theta(t)) \in \mathcal{S}_\delta$. Then, from Definition 7.2.1, for all $t \geq 0$, $\|C_x \theta(t) - z(t)\| \leq \delta$.

9.3.2 Proof of Theorem 7.2.1

Let $(z, \theta) \in \mathcal{S}_\delta$, then from (7.4),

$$\|z - C_x \theta\| \leq \sqrt{\mathcal{F}(z, \theta)} \leq \delta.$$

Let z be a trajectory of Σ' associated to input v and such that $z(0) = z$. Let θ starting at $\theta(0) = \theta$ be given by

$$\dot{\theta}(t) = A\theta(t) + Bu_{\mathcal{F}}(v(t), z(t), \theta(t)).$$

From equation (7.5), we have that $d\mathcal{F}(z(t), \theta(t))/dt \leq 0$. Therefore, for all $t \geq 0$, $(z(t), \theta(t)) \in \mathcal{S}_\delta$. Furthermore, it implies that for all $t \geq 0$, $u_{\mathcal{F}}(v(t), z(t), \theta(t)) \in U$. Thus, θ is a trajectory of Σ which allows to conclude.

9.3.3 Proof of Proposition 7.2.2

First, let us remark that (7.4) clearly holds. Now, consider the interface $u_{\mathcal{F}}(v, z, \theta) = u_{\mathcal{S}_\delta}(v, z, \theta)$. If $Q(z, \theta) \leq 4\nu^2$, then it is clear that (7.5) holds. If $Q(z, \theta) \geq 4\nu^2$, then we can show that

$$\begin{aligned} & \frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) = \\ & = -2(C_x \theta - z)^T v - 2\alpha(C_x \theta - z + 2C_y \theta)^T v + 2(C_x \theta - z)^T C_x (A\theta + Bu_{\mathcal{F}}) + \\ & \quad + 2\alpha(C_x \theta - z + 2C_y \theta)^T (C_x + 2C_y) (A\theta + Bu_{\mathcal{F}}) = \\ & = -2(C_x \theta - z)^T v - \alpha(C_x \theta - z + 2C_y \theta)^T (2v - 2(C_x + 2C_y)(A\theta + Bu_{\mathcal{F}})) + \\ & \quad + 2(C_x \theta - z)^T C_x (A\theta + Bu_{\mathcal{F}}). \end{aligned}$$

Also, we have

$$C_x(A\theta + Bu_{\mathcal{F}}) = C_xA\theta + C_xBu_{\mathcal{F}} = C_y\theta + 0 = C_y\theta$$

and

$$\begin{aligned} (C_x + 2C_y)(A\theta + Bu_{\mathcal{F}}) &= C_xA\theta + C_xBu_{\mathcal{F}} + 2C_yA\theta + 2C_yBu_{\mathcal{F}} \\ &= C_y\theta + 0 + 0 + 2u_{\mathcal{F}} \\ &= C_y\theta + v + \frac{-1 - \alpha}{2\alpha}(C_x\theta - z) - 2C_y\theta \\ &= v + \frac{-1 - \alpha}{2\alpha}(C_x\theta - z) - C_y\theta. \end{aligned}$$

Hence,

$$\begin{aligned} &\frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) = \\ &= -2(C_x\theta - z)^T v - \alpha(C_x\theta - z + 2C_y\theta)^T (2v - 2v + \frac{1 + \alpha}{\alpha}(C_x\theta - z) + 2C_y\theta) + \\ &\quad + 2(C_x\theta - z)^T C_y\theta = \\ &= -2(C_x\theta - z)^T v - \alpha(C_x\theta - z + 2C_y\theta)^T (C_x\theta - z + 2C_y\theta) - \\ &\quad - \alpha(C_x\theta - z + 2C_y\theta)^T \frac{1}{\alpha}(C_x\theta - z) + 2(C_x\theta - z)^T C_y\theta = \\ &= -2(C_x\theta - z)^T v - \alpha \|C_x\theta - z + 2C_y\theta\|^2 - (C_x\theta - z)^T (C_x\theta - z) - \\ &\quad - 2(C_y\theta)^T (C_x\theta - z) + 2(C_x\theta - z)^T C_y\theta = \\ &= -Q(z, \theta) - 2(C_x\theta - z) \cdot v \leq -Q(z, \theta) + 2\nu \|C_x\theta - z\| \end{aligned}$$

because

$$-2(C_x\theta - z) \cdot v \leq |2(C_x\theta - z) \cdot v| \leq 2\|C_x\theta - z\| \|v\| \leq 2\nu \|C_x\theta - z\|.$$

Since $\|C_x\theta - z\|^2 \leq Q(z, \theta)$, we have

$$\begin{aligned} & \frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) \leq \\ & \leq -Q(z, \theta) + 2\nu\sqrt{Q(z, \theta)} \leq \sqrt{Q(z, \theta)}(2\nu - \sqrt{Q(z, \theta)}). \end{aligned}$$

Since $Q(z, \theta) \geq 4\nu^2$, equation (7.5) holds and \mathcal{F} is a simulation function of Σ' by Σ , and $u_{\mathcal{F}}$ is an associated interface.

Moreover, for all $v \in V$, $(z, \theta) \in \mathcal{S}_\delta$, the interface $u_{\mathcal{F}}(v, z, \theta)$ satisfies the velocity constraints of Σ :

$$\begin{aligned} \|u_{\mathcal{F}}\| &= \|u_{\mathcal{S}_\delta}\| = \left\| \frac{v}{2} + \frac{-1 + \alpha - 2\alpha}{4\alpha} (C_x\theta - z) - C_y\theta \right\| \\ &= \left\| \frac{v}{2} + \frac{-1 + \alpha}{4\alpha} (C_x\theta - z) - \frac{1}{2} (C_x\theta - z + 2C_y\theta) \right\| \\ &\leq \left\| \frac{v}{2} \right\| + \left| \frac{-1 + \alpha}{4\alpha} \right| \|C_x\theta - z\| + \frac{1}{2} \|C_x\theta - z + 2C_y\theta\| \\ &\leq \frac{\nu}{2} + \frac{|-1 + \alpha|}{4\alpha} \sqrt{\mathcal{F}(z, \theta)} + \frac{1}{2} \sqrt{\frac{\mathcal{F}(z, \theta)}{\alpha}} \end{aligned}$$

since $\|C_x\theta - z\|^2 \leq \mathcal{F}(z, \theta)$ and $\alpha\|C_x\theta - z + 2C_y\theta\|^2 \leq \mathcal{F}(z, \theta)$. But, $(z, \theta) \in \mathcal{S}_\delta$, that is $\mathcal{F}(z, \theta) \leq 4\nu^2$, and, hence, using (7.6):

$$u_{\mathcal{F}} \leq \frac{\nu}{2} + \frac{|-1 + \alpha|}{4\alpha} 2\nu + \frac{1}{2} \frac{2\nu}{\sqrt{\alpha}} \leq \frac{\nu}{2} (1 + |1 - 1/\alpha| + 2/\sqrt{\alpha}) \leq \mu.$$

Therefore, Theorem 7.2.1 applies and \mathcal{S}_δ is an approximate simulation relation of precision 2ν of Σ' by Σ and an associated interface is given by $u_{\mathcal{S}_\delta}(v, z, \theta) = u_{\mathcal{F}}(v, z, \theta)$.

9.3.4 Proof of Proposition 7.4.1

Let r be an accepting run of \mathcal{A} . Then by definition, $pr \circ r$ is an accepting run of \mathcal{D}' . By construction of \mathcal{D}' , the input trace $\bar{\gamma}$ that corresponds to the run $pr \circ r$ is $\bar{\gamma} = h_{\mathcal{D}} \circ (pr \circ r)|_1$. Therefore, $\bar{\gamma}$ is a trace of \mathcal{D} and $\gamma = (pr \circ r)|_1$ is a path on \mathcal{D} .

9.3.5 Proof of Theorem 7.4.1

For any $\gamma \in \Gamma$ we prove that if $\langle\langle \phi', \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$, then $\langle\langle \phi', \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$ for any trajectory z of the system $[\Sigma', H'_{\phi'}(\gamma, i)]$ starting at time $t \in \mathbb{R}_{\geq 0}$ and at position $z(t) \in \overline{T^{-1}(\gamma(i))}$.

Case $\phi' = \xi$: We have $\langle\langle \xi, \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$ or by definition $\gamma(i) \in \mathcal{O}_D(\xi)$. That is, $\overline{T^{-1}(\gamma(i))} \subseteq \mathcal{O}_\delta^e(\xi)$ (note that by def. $\mathcal{O}_\delta^e(\xi) = \overline{\mathcal{O}_\delta^e(\xi)}$). Since for some $t \in R$, $z(t) \in \overline{T^{-1}(\gamma(i))}$, we get that $z(t) \in \mathcal{O}_\delta^e(\xi)$. Hence, $\langle\langle \xi, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$.

Conjunction: Let $\phi' = \phi_1 \wedge \phi_2$. By definition, $\langle\langle \phi_1, \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$ and $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$. Moreover, we set $H'_{\phi_1}(\gamma) = H'_{\phi'}(\gamma)$ and $H'_{\phi_2}(\gamma) = H'_{\phi'}(\gamma)$. By the induction hypothesis, for any trajectory z of Σ' starting at some time $t \in R$ at position $z(t) \in \overline{T^{-1}(\gamma(i))}$ and then applying controller $H'_{\phi'}(\gamma, i)$, we get that $\langle\langle \phi_1, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$ and $\langle\langle \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$. Therefore, for any trajectory z of Σ' starting at some time $t \in R$ at position $z(t) \in \overline{T^{-1}(\gamma(i))}$ and then applying controller $H'_{\phi'}(\gamma, i)$, we get that $\langle\langle \phi', \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$.

Disjunction: Let $\phi' = \phi_1 \vee \phi_2$. By definition, we have that $\langle\langle \phi_1, \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$ or $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, i) = \top$. Moreover, we set $H'_{\phi_1}(\gamma) = H'_{\phi'}(\gamma)$ or $H'_{\phi_2}(\gamma) = H'_{\phi'}(\gamma)$. By the induction hypothesis, for any trajectory z of Σ' starting at some time $t \in R$ at position $z(t) \in \overline{T^{-1}(\gamma(i))}$ and then applying controller $H'_{\phi_1}(\gamma, i)$, we get that $\langle\langle \phi_1, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$, or for any trajectory z of Σ' starting at some time $t \in R$ at position $z(t) \in \overline{T^{-1}(\gamma(i))}$ and then applying controller $H'_{\phi_2}(\gamma, i)$, we get that

$\langle\langle\phi_2, \mathcal{O}_\delta^e\rangle\rangle_C(z, t) = \top$. Therefore, for any trajectory z of Σ' starting at some time $t \in R$ at position $z(t) \in \overline{T^{-1}(\gamma(i))}$ and then applying controller $H'_{\phi'}(\gamma, i)$, we get that $\langle\langle\phi', \mathcal{O}_\delta^e\rangle\rangle_C(z, t) = \top$.

Until: Let $\phi' = \phi_1 \mathcal{U} \phi_2$. There exists some $j \geq i$ such that $\langle\langle\phi_2, \mathcal{O}_D\rangle\rangle_D(\gamma, j) = \top$ and for all $k \in [i, j)$ we get that $\langle\langle\phi_1, \mathcal{O}_D\rangle\rangle_D(\gamma, k) = \top$. We set $H'_{\phi_2}(\gamma) = H'_{\phi_1}(\gamma) = H'_{\phi'}(\gamma)$. Consider now the trajectory z that is generated by Σ' using the controller $H'_{\phi'}(\gamma, i)$. The initial condition for the trajectory z is any point in the initial cell, i.e., $z(t) \in \overline{T^{-1}(\gamma(i))}$. By construction, there exists a sequence of times $t = t_i \leq t_{i+1} \leq \dots \leq t_j$, where t_k for $k = i + 1, \dots, j$ is the time that the trajectory z crosses the edge $\overline{T^{-1}(\gamma(k-1))} \cap \overline{T^{-1}(\gamma(k))}$. Consider any time instant $t'' \in [t_k, t_{k+1}]$ for any $k \in [i, j)$. Then, we know that $z(t'') \in \overline{T^{-1}(\gamma(k))}$. Now the induction hypothesis applies and applying the hybrid controller $H'_{\phi_1}(\gamma, k)$ we get that $\langle\langle\phi_1, \mathcal{O}_\delta^e\rangle\rangle_C(z, t'') = \top$. Therefore, for all $t'' \in [t, t_j]$, we have $\langle\langle\phi_1, \mathcal{O}_\delta^e\rangle\rangle_C(z, t'') = \top$. Now, note that $z(t_j) \in \overline{T^{-1}(\gamma(j-1))} \cap \overline{T^{-1}(\gamma(j))}$. Hence, by the induction hypothesis, using controller $H'_{\phi_2}(\gamma, j)$ we get that $\langle\langle\phi_2, \mathcal{O}_\delta^e\rangle\rangle_C(z, t_j) = \top$. Thus, if we set $t' = t_j$, then we are done and $\langle\langle\phi_1 \mathcal{U} \phi_2, \mathcal{O}_\delta^e\rangle\rangle_C(z, t) = \top$. Note that if $\langle\langle\phi_2, \mathcal{O}_D\rangle\rangle_D(\gamma, i) = \top$, then we are immediately done since the induction hypothesis applies directly.

Release: Let $\phi' = \phi_1 \mathcal{R} \phi_2$. Then for all $j \geq i$ we have $\langle\langle\phi_2, \mathcal{O}_D\rangle\rangle_D(\gamma, j) = \top$ or there exists $k \in [i, j)$ such that $\langle\langle\phi_1, \mathcal{O}_D\rangle\rangle_D(\gamma, k) = \top$. We set $H'_{\phi_2}(\gamma) = H'_{\phi_1}(\gamma) = H'_{\phi'}(\gamma)$. Consider now the trajectory z that is generated by Σ' using the controller $H'_{\phi'}(\gamma, i)$. The initial condition for the trajectory z is any point in the initial cell, i.e., $z(t) \in \overline{T^{-1}(\gamma(i))}$. By construction, there exists a sequence of times $t = t_i \leq t_{i+1} \leq \dots \leq t_j$, where t_k for $k = i + 1, \dots, j$ is the time that the trajectory z crosses the edge $\overline{T^{-1}(\gamma(k-1))} \cap \overline{T^{-1}(\gamma(k))}$. There exist two cases now.

First, assume that $\langle\langle\phi_1, \mathcal{O}_D\rangle\rangle_D(\gamma, k) = \top$ is true for some $k \geq i$. The corresponding controller is $H'_{\phi_1}(\gamma, k)$. Since there exists some time instant $t'' \in [t_k, t_{k+1}]$ such that

$z(t'') \in \overline{T^{-1}(\gamma(k))}$, we get by the induction that $\langle\langle \phi_1, \mathcal{O}_\delta^e \rangle\rangle_C(z, t'') = \top$. Note that for all $k' \in [i, k]$, it must also be $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, k') = \top$ (it is easy to see that by the recursive formulation of until in Section 3.2.3). Consider any time instant $t' \in [t_{k'}, t_{k'+1}]$ for any $k' \in [i, k]$. Then, we know that $z(t') \in \overline{T^{-1}(\gamma(k'))}$. Now, the induction hypothesis applies and we get that $\langle\langle \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t') = \top$ using controller $H'_{\phi_2}(\gamma, k')$. Therefore, for all $t' \in [t, t_{k+1}]$ we have $\langle\langle \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t') = \top$. We conclude that $\langle\langle \phi_1 \mathcal{R} \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$.

Second, assume that the solution path γ is given by the first part of the definition, that is by “for all $j \geq i$ we have $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, j) = \top$ ” . When we generate the continuous trajectory using the corresponding hybrid controller $H'_{\phi_2}(\gamma, j)$, we get a total order on the times that the trajectory z crosses each edge, i.e., $t_{i+1} \leq t_{i+2} \leq t_{i+3} \leq \dots$. Using the same reasoning as in the first case we get that for all $t' \geq t$, we have $\langle\langle \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t') = \top$.

Finally, we need to consider the case for a path γ whose unique periodic state repeats indefinitely. This case implies that the specification requires that for all $j \geq i$ we have $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, j) = \top$, which is part of the semantics of the release operator for a formula $\phi' = \phi_1 \mathcal{R} \phi_2$. Here, ϕ_2 is a Boolean combination of propositions from Ξ_{AP} . The hybrid controller $H'_{\phi'}(\gamma, i)$ realizes the “for all $j \geq i$ we have $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, j) = \top$ ” part of the semantics by composing a cell invariant controller. Therefore, for all $t' \geq t$, we get that $z(t') \in \overline{T^{-1}(\gamma(j))}$ for any $j \geq i$ and we conclude that $\langle\langle \phi_2, \mathcal{O}_D \rangle\rangle_D(\gamma, j) = \top$. Hence, $\langle\langle \phi_1 \mathcal{R} \phi_2, \mathcal{O}_\delta^e \rangle\rangle_C(z, t) = \top$.

Remark 9.3.1. *Consider the release case in the above theorem (see proof in Section 9.3.5). If we had not required the existence of at least one cell invariant controller in the periodic part of γ , then potentially this time sequence could have collapsed to a single point in time, i.e., for some $j \geq i$, $t_{j+1} = t_{j+2} = t_{j+3} = \dots$. This would have been undesirable since the finite variability assumption would have been violated and,*

then, we would have introduced in our system Zeno behavior [131].

Nomenclature

Logic

$\langle\langle \cdot, \cdot \rangle\rangle_C$ Continuous-time Boolean semantics of a temporal logic formula, page 23

$\langle\langle \cdot, \cdot \rangle\rangle_D$ Discrete-time Boolean semantics of a temporal logic formula, page 26

$\llbracket \cdot, \cdot \rrbracket_C$ Continuous-time robust semantics of a temporal logic formula, page 32

$\llbracket \cdot, \cdot \rrbracket_D$ Discrete-time robust semantics of a temporal logic formula, page 42

\neg Logical negation, page 21

\vee Disjunction, page 21

\wedge Conjunction, page 21

$+_R$ Addition of a point with a set, intersected with the set R , i.e., $t +_R \mathcal{I} := (t + \mathcal{I}) \cap R$, page 22

\diamond Eventually, page 21

\square Always, page 21

AP A set of atomic propositions, page 20

\mathbb{C} A set of truth degree constants. Usually, $\mathbb{C} = \mathbb{B}$ or $\mathbb{C} = \overline{\mathbb{R}} \cup \mathbb{B}$, page 20

- c A logical constant, page 21
- dur** A function that computes the maximum time for which requirements are imposed by the temporal operators of an MTL formula, page 80
- \mathcal{I} Timing constraints on temporal operators, page 20
- K_{\in} The characteristic function of the \in relation, page 23
- K_{\in}^{∞} A modified version of the characteristic function of the \in relation, page 44
- $\mathcal{L}_t(\phi, \mathcal{O})$ The set of all continuous-time signals that satisfy ϕ starting at time t , page 23
- $\mathcal{L}(\phi, \mathcal{O})$ The set of all continuous-time signals that satisfy ϕ starting at time 0, page 23
- $\hat{\mathcal{L}}_i(\phi, \mathcal{O})$ The set of all discrete-time signals which satisfy an LTL formula ϕ , page 27
- $\mathcal{L}_i^{\tau}(\phi, \mathcal{O})$ The set of all discrete-time signals which when paired with τ satisfy ϕ , page 26
- mtc** A translation function that maps the temporal operators of an MTL formula to the corresponding matching temporal operators, page 85
- $MTL_{\mathbb{C}}(AP)$ The set of MTL formulas built upon AP with constants from \mathbb{C} , page 21
- $MTL_{\mathbb{C}}^{+}(AP)$ The set of MTL formulas built upon AP with constants from \mathbb{C} in Negation Normal Form, page 21
- $MTL_{\mathbb{C}}(AP, op_1, op_2, \dots)$ The set of MTL formulas built upon AP with constants from \mathbb{C} using the logical operators op_1, op_2, \dots , page 21
- $MITL_{\mathbb{C}}(AP)$ The set of MITL formulas built upon AP with constants from \mathbb{C} , page 21

- $LTL_{\mathbb{C}}(AP)$ The set of LTL formulas built upon AP with constants from \mathbb{C} , page 21
- nnf** A function that translates a formula ϕ into NNF, page 27
- \mathcal{O} Observation map, page 22
- \mathcal{O}^{-1} The inverse of an observation map, page 22
- \mathcal{O}^e The extended observation map, page 35
- $\mathcal{O}_\varepsilon^e$ The ε -robustified extended observation map, page 35
- $\mathcal{I}(\phi)$ The set of all timing constraints \mathcal{I} that appear in the temporal operators of an MTL formula ϕ , page 80
- p, p_1, p_2, \dots Atomic propositions, page 21
- $\phi, \phi_1, \phi_2, \psi, \dots$ Temporal logic formulas, page 21
- pos** A translation function that replaces negated atomic propositions by new symbols, page 34
- \mathcal{R} Release, page 21
- $\overline{\mathcal{R}}$ Matching release, page 85
- str $_{\Delta\hat{\tau}}$** A translation function that strengthens the timing constraints of the temporal operators of an MTL formula by $\Delta\hat{\tau}$, page 79
- $TSS_i(\phi, \mathcal{O})$ The set of all timed state sequences that satisfy ϕ starting at time i , page 26
- $TSS(\phi, \mathcal{O})$ The set of all timed state sequences that satisfy ϕ starting at time 0, page 26

$TSS_i^\tau(\phi, \mathcal{O})$ The set of all timed state sequences that satisfy ϕ under a given timing function τ , page 26

\mathcal{U} Until, page 21

$\bar{\mathcal{U}}$ Matching until, page 85

Ξ_{AP} The extended set of atomic propositions, page 34

Matrices

\mathbf{I}_n The $n \times n$ identity matrix, page 97

Motion Planning

$\rightarrow_{\mathcal{D}}$ The transition relation of \mathcal{D} , page 129

\mathcal{B} A Büchi automaton, page 131

$\mathcal{B}_{\phi'}$ A Büchi automaton that accepts the infinite traces which satisfy the specification ϕ' , page 131

\mathcal{A} The Büchi automaton which corresponds to the product of \mathcal{D}' with $\mathcal{B}_{\phi'}$, page 133

\mathcal{D} A finite transition system, page 129

\mathcal{D}' The Büchi automaton which corresponds to the FTS \mathcal{D} , page 132

$F_{\mathcal{B}}$ The set of accepting states of the Büchi automaton \mathcal{B} , page 131

γ A path on the FTS \mathcal{D} , page 130

$\tilde{\gamma}$ The sequence of observations generated by a path γ , page 130

γ^f The non-periodic part of a path γ , page 135

- γ^l The periodic part of a path γ , page 135
- $h_{\mathcal{D}}$ A map from equivalence classes to atomic propositions, page 129
- h_{δ} A mapping from the modified workspace of the robot to the extended set of atomic propositions, page 126
- $H'_{\phi'}(\gamma)$ The hybrid controller which corresponds to the path γ that satisfies formula ϕ' , page 139
- $H'_{\phi'}(\gamma, i)$ The hybrid controller which corresponds to the path γ that satisfies formula ϕ' starting at position i of the path, page 139
- $\lambda_{\mathcal{B}}$ The transition relation of the Büchi automaton \mathcal{B} , page 131
- μ The bound on the control input of the dynamics model of the robot, page 117
- ν The bound on the control input to the kinematics model of the robot, page 120
- Ω An input alphabet, page 131
- \mathcal{O}_D An observation map that maps atomic propositions to equivalence classes, page 132
- \mathcal{Q} The set of equivalence classes of points which satisfy the same set of atomic propositions, page 129
- Q The set of equivalence classes that represent the interior of the cells, page 129
- Q_0 The set of possible initial cells, page 129
- r A run of a Büchi automaton, page 131
- Σ A dynamics model of a mobile robot, page 117

Σ'	A kinematics model of a mobile robot, page 120
$S_{\mathcal{B}}$	The set of states of the Büchi automaton \mathcal{B} , page 131
T	A function which maps each state $z \in Z$ to one equivalence class in \mathcal{Q} , page 129
T^{-1}	A function which maps each equivalence class in \mathcal{Q} to the set of states in Z that it represents, page 129
θ	The position and velocity of the dynamics model of the robot, page 121
u	The control input to the dynamics model of the robot, page 117
v	The control input of the kinematics model of the robot, page 120
X	The free workspace of the dynamics model of the robot, page 117
x	The position of the dynamics model of the robot on the plane, page 117
X_0	The set of possible initial positions of the dynamics model of the robot, page 117
y	The velocity of the dynamics model of the robot, page 121
Z	The free workspace of the kinematics model of the robot, page 120
z	The position of the kinematics model of the robot on the plane, page 120
Z_0	The set of possible initial positions of the kinematics model of the robot, page 120

Other Symbols

dom (f)	The domain of a function f , page 17
id	The identity function, page 65

◦ Function composition : $(f \circ g)(t) = f(g(t))$, page 19

Sets and Metric Spaces

$B_d(x, \varepsilon)$ The ε -neighborhood of point x under the metric d , page 17

\mathbb{B} The set of Boolean constants, page 17

$C_d(S, \varepsilon)$ Contraction of a set S by ε under the metric d , page 35

d A metric. Usually, the metric that induces the topology of X , page 15

d_1 Manhattan distance, page 46

d_d The discrete metric, page 18

depth $_d(x, S)$ The depth of point x in a set S under the metric d , page 16

d_e Euclidean distance, page 46

Dist $_d(x, S)$ The signed distance of point x from a set S under the metric d , page 16

dist $_d(x, S)$ The distance of point x from a set S under the metric d , page 16

$\mathcal{EP}(P)$ The set of extreme points of a convex set P , page 99

$E_d(S, \varepsilon)$ Expansion of a set S by ε under the metric d , page 35

\mathbb{N} The set of natural numbers, page 19

$\mathcal{P}(S)$ The powerset of the set S , page 22

\mathbb{Q} The set of rational numbers, page 83

\mathbb{R} The reals, page 16

$\overline{\mathbb{R}}$ The extended reals $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$, page 16

- S A set. Usually, $S \subseteq X$, page 16
- \bar{S} The closure of a set S , page 16
- X A metrizable topological space, page 15

Signals

- $\Delta\hat{\tau}$ The maximum sampling step, page 78
- $\delta\tau$ A function that returns the time difference between the current and the previous samples, page 44
- \mathcal{E} A function that bounds the dynamics of a continuous-time signal, page 78
- $\mathfrak{F}(A, B)$ The set of all functions from A to B , page 17
- $\mathfrak{F}(R, X)$ The set of all continuous-time signals from R to X , page 17
- $\mathfrak{F}(N, X)$ The set of all discrete-time signals from N to X , page 19
- $\mathfrak{F}(N, X) \times \mathfrak{F}^\dagger(N, \mathbb{R}_{\geq 0})$ The set of all timed state sequences, page 19
- μ A timed state sequence, i.e., $\mu = (\sigma, \tau)$, page 19
- $\mu^{(1)}$ The discrete-time signal, i.e., σ , of a timed state sequence, page 19
- $\mu^{(2)}$ The timing function, i.e., τ , of a timed state sequence, page 19
- R The domain of a continuous-time signal, $R = [0, r]$ with $r > 0$ or $R = \mathbb{R}_{\geq 0}$, page 17
- ρ The supremum metric for continuous-time signals, page 29
- $\hat{\rho}$ The supremum metric for discrete-time signals, page 39

- s A continuous-time signal, page 17
- σ A discrete-time signal, page 18
- $\mathfrak{F}^\dagger(N, \mathbb{R})$ The set of timing functions, page 19
- $\mathfrak{F}_c^\dagger(N, \mathbb{R})$ The set of timing functions with constant time step, page 19
- $\mathfrak{F}^\dagger(N, R)$ The set of sampling functions, page 20
- τ A timing function, page 18
- τ^{-1} The inverse of timing or sampling function, page 25
- $\hat{\tau}$ A sampling function, page 20

Systems

- \preceq_δ A δ -approximate simulation relation between systems, page 68
- \sim_δ A δ -approximate bisimulation relation between systems, page 68
- Σ A dynamical system, page 60
- Σ^{LTI} A Linear Time Invariant (LTI) system, page 62
- Σ^{LTV} A Linear Time Varying (LTV) system, page 62
- Σ^{LPV} A Linear Parameter Varying (LPV) system, page 62
- $\Sigma^{\text{LPV}}(p_0)$ An LTI system that is derived from an LPV system for a specific constant specific constant parameter value $p_0 \in P$, page 62
- $\bar{\Sigma}$ A closed-loop dynamical system, page 63
- $\bar{\Sigma}^{\text{LTI}}$ A closed-loop Linear Time Invariant (LTI) system, page 63

- $\overline{\Sigma}^{\text{LTV}}$ A closed-loop Linear Time Varying (LTV) system, page 63
- $\overline{\Sigma}^{\text{LPV}}$ A closed-loop Linear Parameter Varying (LPV) system, page 63
- $\overline{\Sigma}_*^{\text{LPV}}$ A regular LPV system, page 95
- \mathcal{B}_δ A δ -approximate bisimulation relation, page 68
- f System dynamics, page 60
- g Observation function, page 60
- $\Lambda(\Sigma)$ Internal language of Σ : the set of state trajectories of system Σ , page 63
- $\mathcal{L}(\Sigma)$ Language of Σ : the set of observable trajectories or traces of system Σ , page 63
- P Parameter space, page 60
- p A parameter function that takes values in P , page 61
- \mathcal{S}_δ A δ -approximate simulation relation, page 68
- \mathbb{T} Time domain, page 60
- $TSS^\tau(\Sigma)$ The set of timed state sequences of Σ that result by pairing the timing or sampling function τ with the (sampled) observable trajectories of system Σ , page 63
- U Input space, page 60
- u An input function that takes values in U , page 61
- X State space, page 60
- x State trajectory, page 61

- X^0 Initial conditions, page 60
- $x^{(i)}$ The i -th component of the state vector $x(t)$, page 65
- Y Observation space, page 60
- y Observable trajectory, page 61
- $y^{(i)}$ The i -th component of the observation vector $y(t)$, page 65

Bibliography

- [1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In John Mitchell, editor, *5th Annual IEEE Symp. on Logic in Computer Science (LICS)*, pages 414–425. IEEE Computer Society Press, June 1990.
- [2] R. Alur, A. Das, J.M. Esposito, R. Fierro, Y. Hur, G. Grudic, V. Kumar, I. Lee, J. P. Ostrowski, G. J. Pappas, J. Southall, J. Spletzer, and C. Taylor. A framework and architecture for multi-robot coordination. *International Journal of Robotics Research*, 21(10-11):977–995, 2002.
- [3] R. Alur, T. A. Henzinger, G. Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(2):971–984, 2000.
- [4] Rajeev Alur and David L Dill. Theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
- [5] Rajeev Alur, Tomas Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.
- [6] Rajeev Alur and Thomas A. Henzinger. Real-Time Logics: Complexity and Expressiveness. In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 390–401, Washington, D.C., 1990. IEEE Computer Society Press.

- [7] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng.*, 22(3):181–201, 1996.
- [8] Rajeev Alur and George J. Pappas, editors. *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, volume 2993 of *Lecture Notes in Computer Science*. Springer, 2004.
- [9] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 291–302, 2001.
- [10] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Perturbed timed automata. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 70–85, 2005.
- [11] Francesco Amato. *Robust Control of Linear Systems Subject to Uncertain Time-Varying Parameters*. Springer, 2006.
- [12] Marco Antoniotti and Bud Mishra. Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1441–1446, May 1995.
- [13] P.J. Antsaklis, editor. *Special issue on hybrid systems: theory and applications*, volume 88(7) of *Proceedings of the IEEE*. IEEE, July 2000.
- [14] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise linear dynamical systems. In *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, pages 21–31. Springer, 2000.

- [15] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In P. Antsaklis et al, editor, *Hybrid Systems II*, volume 999 of *LNCS*, pages 1–20. Springer, 1995.
- [16] M. Barbeau, F. Kabanza, and R. St-Denis. Synthesizing plant controllers using real-time goals. In *Proceedings of 14th International Joint Conference on Artificial Intelligence*, pages 791–798. Morgan Kaufmann, 1995.
- [17] Luciano Baresi and Mauro Pezzè. An introduction to software testing. *Electr. Notes Theor. Comput. Sci.*, 148(1):89–111, 2006.
- [18] Selcuk Bayraktar, Georgios E. Fainekos, and George J. Pappas. Experimental cooperative control of fixed-wing unmanned aerial vehicles. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 4292–4298, December 2004.
- [19] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim Guldstrand Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In Werner Damm and Holger Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification*, volume 4590 of *LNCS*, pages 121–125. Springer, 2007.
- [20] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems : 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, number 3185 in *LNCS*, pages 200–236. Springer, September 2004.

- [21] Calin Belta, Volkan Isler, and George J. Pappas. Discrete abstractions for robot motion planning and control. *IEEE Transactions on Robotics*, 21(5):864–874, October 2005.
- [22] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, and Philippe Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer, 2001.
- [23] Antonia Bertolino. Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE)*, pages 85–103, Washington, DC, USA, 2007. IEEE Computer Society.
- [24] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific, 2 edition, 2000.
- [25] Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust model-checking of linear-time properties in timed automata. In Jose R. Correa, Alejandro Hevia, and Marcos Kiwi, editors, *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, volume 3887 of *LNCS*, pages 238–249, Valdivia, Chile, March 2006. Springer.
- [26] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004.
- [27] John P. Burgess and Yuri Gurevich. The decision problem for linear temporal logic. *Notre Dame Journal of Formal Logic*, 26(2):115–128, April 1985.
- [28] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, June 1999.

- [29] T. D. Burton. *Introduction to Dynamic Systems Analysis*. McGraw-Hill, 1994.
- [30] B. Caillaud, P. Darondeau, L. Lavagno, and X. Xie, editors. *Synthesis and Control of Discrete Event Systems*. Kluwer Academic Press, 2002.
- [31] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [32] Marsha Chechik, Benet Devereux, Steve Easterbrook, and Arie Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology*, 12(4):1–38, October 2004.
- [33] Marsha Chechik, Benet Devereux, and Arie Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In *8th International SPIN Workshop*, volume 2057 of *LNCS*, pages 16–36. Springer, 2001.
- [34] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 3rd edition, 1998.
- [35] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms and Implementations*. MIT Press, March 2005.
- [36] A. Chutinan and B. H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 2, pages 2089–2094, December 1998.
- [37] A. Chutinan and B.H. Krogh. Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 76–90. Springer, 1999.

- [38] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Proceedings of International Conference on Computer-Aided Verification*, July 2002.
- [39] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D.E. Long, K.L. McMillan, and L.A. Ness. Verification of the Futurebus+ Cache Coherence Protocol. In D. Agnew, L. Claesen, and R. Camposano, editors, *The Eleventh International Symposium on Computer Hardware Description Languages and their Applications*, pages 5–20, Ottawa, Canada, 1993. Elsevier.
- [40] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions On Programming Languages and Systems*, 8(2):244–263, 1986.
- [41] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, Massachusetts, 1999.
- [42] David C Conner, Howie Choset, and Alfred Rizzi. Integrated planning and control for convex-bodied nonholonomic systems using local feedback control policies. In *Proceedings of Robotics: Science and Systems II*, Cambridge, USA, June 2006.
- [43] David C Conner, Alfred Rizzi, and Howie Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3546–3551, 2003.

- [44] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms*. MIT Press/McGraw-Hill, second edition, September 2001.
- [45] Thao Dang. *Verification and Synthesis of Hybrid Systems*. PhD thesis, INPG, 2000.
- [46] Thao Dang. Approximate reachability computation for polynomial systems. In João P. Hespanha and Ashish Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *LNCS*, pages 138–152. Springer, 2006.
- [47] Tathagato Rai Dastidar and P. P. Chakrabarti. A verification system for transient response of analog circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):1–39, 2007.
- [48] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, United Kingdom, 2 edition, 2002.
- [49] G. N. Davrazos and N. T. Koussoulas. A review of stability results for switched and hybrid systems. In *Proceedings of the 9th Mediterranean Conference on Control and Automation*, June 2001.
- [50] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2988 of *LNCS*, pages 77–92. Springer, 2004.
- [51] Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching metrics for quantitative transition systems. In *Proceedings of the 31st ICALP*, volume 3142 of *LNCS*, pages 97–109. Springer, 2004.

- [52] Luca de Alfaro and Zohar Manna. Verification in continuous time by discrete reasoning. In *Proceedings of the 4th AMAST*, volume 936 of *LNCS*, pages 292–306. Springer, 1995.
- [53] Mark de Berg, Otfried Schwarzkopf, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.
- [54] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 174–189. Springer, 2007.
- [55] Geir E. Dullerud and Fernando Paganini. *A Course in Robust Control Theory : A Convex Approach*. Springer, 2000.
- [56] E. Allen Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.
- [57] Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2004.
- [58] Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 2007. (Accepted).
- [59] Georgios E. Fainekos, Antoine Girard, and George J. Pappas. Temporal logic verification using simulation. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems*, volume 4202 of *LNCS*, pages 171–186. Springer, 2006.

- [60] Georgios E. Fainekos, Antoine Girard, and George J. Pappas. Hierarchical synthesis of hybrid controllers from temporal logic specifications. In *Hybrid Systems: Computation and Control*, number 4416 in LNCS, pages 203–216. Springer, 2007.
- [61] Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Hybrid controllers for path planning: A temporal logic approach. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 4885 – 4890, December 2005.
- [62] Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2032–2037, April 2005.
- [63] Georgios E. Fainekos, Savvas G. Loizou, and George J. Pappas. Translating temporal logic to controller specifications. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 899–904, December 2006.
- [64] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications. In *Formal Approaches to Testing and Runtime Verification*, volume 4262 of LNCS, pages 178–192. Springer, 2006.
- [65] Georgios E. Fainekos and George J. Pappas. Robust sampling for MITL specifications. In Jean-Francois Raskin and P. S. Thiagarajan, editors, *Proceedings of the 5th International Conference on Formal Modelling and Analysis of Timed Systems*, volume 4763 of LNCS, pages 147–162. Springer, 2007.
- [66] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous time signals. *Theoretical Computer Science*, 2007. (Accepted).

- [67] Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *Proceedings of the 13th International Workshop and 8th Annual Conference of the EACSL on Computer Science Logic (CSL)*, pages 126–140, London, UK, 1999. Springer-Verlag.
- [68] Emilio Frazzoli. *Robust hybrid control for autonomous vehicle motion planning*. PhD thesis, Massachusetts Institute of Technology, May 2001.
- [69] Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.
- [70] Carsten Fritz. Constructing Büchi automata from LTL using simulation relations for alternating Büchi automata. In *the 8th International Conference on Implementation and Application of Automata*, volume 2759 of *LNCS*, pages 35–48, 2003.
- [71] Carlo A. Furia and Matteo Rossi. Integrating discrete and continuous time metric temporal logics through sampling. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems*, volume 4202 of *LNCS*, pages 215–229. Springer, 2006.
- [72] Carlo A. Furia and Matteo Rossi. On the expressiveness of mtl variants over dense time. In Jean-Francois Raskin and P. S. Thiagarajan, editors, *Proceedings of the 5th International Conference on Formal Modelling and Analysis of Timed Systems*, volume 4763 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2007.

- [73] A. Fusaoka, H. Seki, and K. Takahashi. A description and reasoning of plant controllers in temporal logic. In *Proceedings of 8th International joint Conference on Artificial Intelligence*, pages 405–408, 1983.
- [74] Giuseppe De Giacomo and Moshe Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *European Conference on Planning*, volume 1809 of *LNCS*, pages 226–238. Springer, 1999.
- [75] Dimitra Giannakopoulou and Klaus Havelund. Automata-based verification of temporal properties on running programs. In *Proceedings of the 16th IEEE international conference on Automated software engineering*, 2001.
- [76] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 291–305, 2005.
- [77] Antoine Girard and George J. Pappas. Approximate bisimulations for constrained linear systems. In *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, pages 4700–4705, 2005.
- [78] Antoine Girard and George J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, pages 684–689, 2005.
- [79] Antoine Girard and George J. Pappas. Hierarchical control using approximate simulation relations. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.
- [80] Antoine Girard and George J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control (HSCC)*, volume 3927 of *LNCS*, pages 272 – 286. Springer, 2006.

- [81] Antoine Girard and George J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307–1317, 2007.
- [82] Antoine Girard and George J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Trans. Auto. Cont.*, 52(5):782–798, 2007.
- [83] Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *European Conference on Planning*, volume 1809 of *LNCS*, pages 1–20, 1999.
- [84] Vineet Gupta, Thomas A. Henzinger, and Radha Jagadeesan. Robust timed automata. In *HART '97: Proceedings of the International Workshop on Hybrid and Real-Time Systems*, pages 331–345, London, UK, 1997. Springer-Verlag.
- [85] Luc C.G.J.M. Habets and J. H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.
- [86] Brent Hailpern and Padmanabhan Santhanam. Software debugging, testing, and verification. *IBM Systems Journal*, 41(1):4–12, 2002.
- [87] Z. Han and B. H. Krogh. Reachability analysis of hybrid control systems using reduced-order models. In *Proceedings of the American Control Conference*, pages 1183–1189, 2004.
- [88] Zhi Han. *Formal Verification of Hybrid Systems using Model Order Reduction and Decomposition*. PhD thesis, Dept. of ECE, Carnegie Mellon University, 2005.
- [89] Cheryl Harkness and Elizabeth Wolf. Verifying the summit bus converter protocols with symbolic model checking. *Formal Methods in System Design*, 4(2):83–97, 1994.

- [90] Walter Hartong, Lars Hedrich, and Erich Barke. On discrete modeling and model checking for nonlinear analog systems. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV)*, pages 401–413, London, UK, 2002. Springer-Verlag.
- [91] K. Havelund, M. Lowry, and J. Penix. Formal analysis of a space-craft controller using spin. *IEEE Trans. Software Eng.*, 27:749 – 765, 2001.
- [92] Klaus Havelund and Grigore Rosu. Monitoring programs using rewriting. In *Proceedings of the 16th IEEE international conference on Automated software engineering*, 2001.
- [93] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
- [94] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. In *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *LNCS*, pages 460–463. Springer, 1997.
- [95] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What’s decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.
- [96] Thomas A. Henzinger, Rupak Majumdar, and Vinayak S. Prabhu. Quantifying similarities between timed systems. In *FORMATS*, volume 3829 of *LNCS*, pages 226–241. Springer, 2005.
- [97] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In *Proceedings of the 19th ICALP*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.

- [98] Thomas A. Henzinger and Jean-François Raskin. Robust undecidability of timed and hybrid systems. In Lynch and Krogh [133], pages 145–159.
- [99] Yoram Hirshfeld and Alexander Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inf.*, 62(1):1–28, 2004.
- [100] G.J. Holzmann. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, Reading, Massachusetts, 2004.
- [101] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 2 edition, 2001.
- [102] H. Horisberger and P. Belanger. Regulators for linear, time invariant plants with uncertain parameters. *IEEE Trans. Auto. Cont.*, 21(5):705–708, 1976.
- [103] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
- [104] D. Hristu-Varsakelis, M. Egerstedt, and P. S. Krishnaprasad. On the complexity of the motion description language MDLe. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3360–3365, December 2003.
- [105] Jinfeng Huang, Jeroen Voeten, and Marc Geilen. Real-time property preservation in approximations of timed systems. In *Proceedings of the 1st ACM & IEEE International Conference on Formal Methods and Models for Co-Design*, pages 163–171, 2003.
- [106] C. S. Indulkar. State space analysis of a ladder network representing a transmission line. *International Journal of Electrical Engineering Education*, 42(4):383–392, 2005.

- [107] A. Agung Julius, Georgios E. Fainekos, Madhukar Anand, Insup Lee, and George J. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control*, number 4416 in LNCS, pages 329–342. Springer, 2007.
- [108] Froduald Kabanza. Synchronizing multiagent plans using temporal logic specifications. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, pages 217–224, San Francisco, CA, 1995. MIT Press.
- [109] Jim Kapinski, Bruce H. Krogh, Oded Maler, and Olaf Stursberg. On systematic simulation of open continuous systems. In *Hybrid Systems: Computation and Control*, volume 2623 of LNCS, pages 283–297. Springer, 2003.
- [110] J. M. Keil. Polygon decomposition. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 491–518. Elsevier Science, North-Holland, Amsterdam, 2000.
- [111] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, second edition, 1996.
- [112] Eric Klavins. Automatic compilation of concurrent hybrid factories from product assembly specifications. In *Hybrid Systems: Computation and Control*, volume 1790 of LNCS, pages 174–187. Springer, 2000.
- [113] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from LTL specifications. In *Proceedings of Hybrid Systems: Computation and Control*, volume 3927 of LNCS, pages 333–347. Springer, 2006.
- [114] Marius Kloetzer and Calin Belta. Hierarchical abstractions for robotic swarms. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.

- [115] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7):1026 – 1049, July 2000.
- [116] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
- [117] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. From structured english to robot motion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2722, October 2007.
- [118] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Where’s Waldo? Sensor-Based Temporal Logic Motion Planning. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 3116–3121, April 2007.
- [119] Kåre J. Kristoffersen, Christian Pedersen, and Henrik Reif Andersen. Run-time verification of timed LTL using disjunctive normalized equation systems. In *Proceedings of the 3rd Workshop on Run-time Verification*, volume 89 of *ENTCS*, pages 1–16, 2003.
- [120] Benjamin C. Kuo. *Digital Control Systems*. Oxford University Press, Inc., New York, NY, USA, 1992.
- [121] Orna Kupferman, P. Madhusudan, P. S. Thiagarajan, and Moshe Y. Vardi. Open systems in reactive environments: Control and synthesis. In *Proceedings of the 11th CONCUR*, volume 1877 of *LNCS*, pages 92–107, 2000.
- [122] A. Kurzanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*. Springer, 2000.

- [123] A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for hybrid dynamics: the reachability problem. In Wijesuriya P. Dayawansa, Anders Lindquist, and Yishao Zhou, editors, *New Directions and Applications in Control Theory*, volume 321 of *LNCIS*, pages 193–205. Springer, 2005.
- [124] Tom Laureys. From event-based semantics to linear temporal logic: The logical and computational aspects of a natural language interface for hardware verification. Master’s thesis, University of Edinburgh, November 1999.
- [125] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- [126] D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, 1999.
- [127] Stephen R. Lindemann and Steven M. LaValle. Smoothly blending vector fields for global robot navigation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3553–3559, December 2005.
- [128] Scott Little, David Walter, Kevin Jones, and Chris J. Myers. Analog/mixed-signal circuit verification using models generated from simulation traces. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 4762 of *LNCS*, pages 114–128. Springer, 2007.
- [129] Savvas G. Loizou and Kostas J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, December 2004.
- [130] Iannelli Luigi, Karl H. Johansson, Ulf Jonsson, and Vasca Francesco. Averaging of nonsmooth systems using dither. *Automatica*, 42(4):669–676, April 2006.

- [131] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48:2–17, 2003.
- [132] J. Lygeros, C. Tomlin, , and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.
- [133] Nancy A. Lynch and Bruce H. Krogh, editors. *Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, Pittsburgh, PA, USA, March 23-25, 2000, Proceedings*, volume 1790 of *LNCS*. Springer, 2000.
- [134] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, volume 3253 of *LNCS*, pages 152–166, 2004.
- [135] Oded Maler, Dejan Nickovic, and Amir Pnueli. On synthesizing controllers from bounded-response properties. In Werner Damm and Holger Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification*, volume 4590 of *LNCS*, pages 95–107. Springer, 2007.
- [136] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems — Specification*. Springer, 1992.
- [137] Zohar Manna and Pierre Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 6(1):68–93, 1984.
- [138] Nicolas Markey and Philippe Schnoebelen. Model checking a path (preliminary report). In *Proceedings of the 14th International Conference on Concurrency Theory*, volume 2761 of *LNCS*, pages 251–265, August 2003.

- [139] A. Martinon. Distance to the intersection of two sets. *Bull. Austral. Math. Soc.*, 70:329341, 2004.
- [140] Otto Mayr. Victorian physicists and speed regulation: An encounter between science and technology. *Notes and Records of the Royal Society of London*, 26(2):205–228, 1971.
- [141] Kenneth Lauchlin McMillan. *Symbolic model checking: an approach to the state explosion problem*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1992.
- [142] Robin Milner. *Communicating and mobile systems: the μ -calculus*. Cambridge University Press, New York, NY, USA, 1999.
- [143] Ian Mitchell and Claire Tomlin. Level set methods for computation in hybrid systems. In Lynch and Krogh [133], pages 310–323.
- [144] Ian M. Mitchell. A toolbox of level set methods. Technical Report TR-2007-11, Department of Computer Science, University of British Columbia, June 2007.
- [145] T. Moor and J. M. Davoren. Robust controller synthesis for hybrid systems using modal logic. In *Proceedings of Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 433–446. Springer, 2001.
- [146] James Munkres. *Topology*. Prentice Hall, 2nd edition, 1999.
- [147] Atul Narkhede and Dinesh Manocha. Fast polygon triangulation based on seidel’s algorithm. In Alan W. Paeth, editor, *Graphics Gems V*, chapter VII.5, pages 394–397. Academic Press, 1995.
- [148] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 4 edition, 2001.

- [149] Peter Ohrstrom and Per F. V. Hasle. *Temporal Logic: From Ancient Ideas to Artificial Intelligence*. Springer, 1995.
- [150] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *20th IEEE Symposium on Logic in Computer Science (LICS)*, pages 188–197, 2005.
- [151] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *Proceedings of the International Conference on Robotics and Automation*, pages 3914–3919, New Orleans, LA, April 2004.
- [152] George J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, December 2003.
- [153] L. Pillage, R. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, 1995.
- [154] Nir Piterman, Amir Pnueli, and Yaniv Sa’ar. Synthesis of Reactive(1) Designs. In *VMCAI*, pages 364–380, Charleston, SC, January 2006.
- [155] Erion Plaku, Lydia E. Kavragi, and Moshe Y. Vardi. Hybrid systems: From verification to falsification. In Werner Damm and Holger Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification*, volume 4590 of *LNCS*, pages 463–476. Springer, 2007.
- [156] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium Foundations of Computer Science*, pages 46–57, 1977.
- [157] Amir Pnueli. Development of hybrid systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *LNCS*, pages 77–85. Springer, 1994.

- [158] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science*, volume II, pages 746–757. IEEE, 1990.
- [159] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 477 – 492. Springer, 2004.
- [160] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.
- [161] Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
- [162] Arthur Quaid and Alfred Rizzi. Robust and efficient motion planning for a planar robot using hybrid control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 4021 – 4026, April 2000.
- [163] Michael M. Quottrup, Thomas Bak, and Roozbeh Izadi-Zamanabadi. Multi-robot planning: A timed automata approach. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4417–4422, April 2004.
- [164] Jean-François Raskin and Pierre-Yves Schobbens. Real-time logics: Fictitious clock as an abstraction of dense time. In Ed Brinksma, editor, *Proceedings of the 3rd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 1217 of *LNCS*, pages 165–182. Springer, 1997.

- [165] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 IEEE American Control Conference*, pages 1936–1941, May 2002.
- [166] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1996.
- [167] S. Sastry. *Nonlinear Systems*. Springer, 2004.
- [168] K. Schneider. *Verification of Reactive Systems – Formal Methods and Algorithms*. Texts in Theoretical Computer Science (EATCS Series). Springer, 2004.
- [169] B. I. Silva and B. H. Krogh. Formal verification of hybrid systems using CheckMate: a case study. In *Proceedings of the American Control Conference*, volume 3, pages 1679 – 1683, June 2000.
- [170] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.
- [171] Paulo Tabuada and George J. Pappas. From discrete specifications to hybrid control. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, December 2003.
- [172] Paulo Tabuada and George J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.
- [173] Li Tan, Jesung Kim, Oleg Sokolsky, and Insup Lee. Model-based testing and monitoring for hybrid embedded systems. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration*, pages 487–492, 2004.

- [174] Prasanna Thati and Grigore Rosu. Monitoring algorithms for metric temporal logic specifications. In *Runtime Verification*, volume 113 of *ENTCS*, pages 145–162. Elsevier, 2005.
- [175] J. G. Thistle and W. M. Wonham. Supervision of infinite behavior of discrete-event systems. *SIAM J. Control Optim.*, 32(4):1098–1113, 1994.
- [176] Stavros Tripakis and Karine Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems*, volume 1708, pages 233–252. Springer, 1999.
- [177] G. R. Wood and B. P. Zhang. Estimation of the Lipschitz constant of a function. *Journal of Global Optimization*, 8(1):91–103, 1996.
- [178] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost asap semantics: from timed models to timed implementations. *Form. Asp. Comput.*, 17(3):319–341, 2005.
- [179] Hakan Yazarel and George J. Pappas. Geometric programming relaxations for linear system reachability. In *Proceedings of the American Control Conference*, June 2004.