Contributions to the Control of Networked Cyber-Physical Systems

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

by

Manuel Mazo Espinosa

2010
The dissertation of Manuel Mazo Espinosa is approved.

________________________________________
Robert T. M’Closkey

________________________________________
Mani B. Srivastava

________________________________________
Panagiotis D. Christofides

________________________________________
Paulo Tabuada, Committee Chair

University of California, Los Angeles
2010
To my family:

*to the new incorporations*

and *to those that left us.*
# Table of Contents

1 Introduction ................................................................. 1

1.1 Motivation ................................................................. 1

1.2 Organization of the thesis ............................................ 4

2 Controller implementations over wireless sensor/actuator networks .............................. 5

2.1 Introduction ................................................................. 5

2.1.1 Previous work ......................................................... 6

2.1.2 Contributions .......................................................... 9

2.2 Preliminaries ............................................................... 11

2.2.1 Notation ................................................................. 11

2.2.2 Mathematical systems theory ..................................... 12

2.3 Event-triggered control .................................................. 16

2.4 Decentralized event-triggered control ................................ 18

2.4.1 Decentralized conditions .......................................... 19

2.4.2 Adaptation .............................................................. 21

2.5 Self-triggered control for linear systems ................................ 23

2.5.1 Implementation ...................................................... 25

2.5.2 Input to state stability ............................................. 27

2.6 Examples ................................................................. 30

2.6.1 Decentralized event-triggered control .......................... 30
3 Correct-by-design synthesis of embedded controllers

3.1 Introduction

3.1.1 Previous work

3.1.2 Contributions

3.2 Preliminaries

3.2.1 Notation

3.2.2 Systems and control systems

3.2.3 Incremental forward completeness

3.2.4 Systems relations

3.2.5 Composition of systems

3.3 Symbolic models for control systems

3.3.1 Existence of symbolic models

3.3.2 Simplifications

3.4 Approximate time-optimal control

3.4.1 Problem definition

3.4.2 Entry time bounds

3.4.3 Controller design

3.4.4 Controller refinement

3.4.5 Approximate time-optimal synthesis in practice
3.5 Pessoa: A Matlab toolbox for the synthesis of correct-by-design embedded controllers ................................. 76
  3.5.1 Binary Decision Diagrams ................................. 76
  3.5.2 Software design as a controller synthesis problem .......... 77
  3.5.3 Computing symbolic models in Pessoa ....................... 78
  3.5.4 Synthesizing symbolic controllers in Pessoa ................. 81
  3.5.5 Simulating the closed-loop in Simulink ..................... 82

3.6 Examples .......................................................... 83
  3.6.1 DC Motor ....................................................... 83
  3.6.2 Control with shared actuators ............................... 86
  3.6.3 Approximate time-optimal control of a double integrator ... 90
  3.6.4 Approximate time-optimal control of a unicycle ............. 93

3.7 Discussion .......................................................... 96

3.8 Appendix: Proofs .................................................... 99

4 Conclusion .............................................................. 105

References ............................................................... 107
List of Figures

2.1 Illustration of a typical WSAN network infrastructure. 19
2.2 The quadruple-tank system. 31
2.3 Times between updates, evolution of the triggering condition, and evolution of the states for the centralized event-triggering implementation. 34
2.4 Times between updates, evolution of the triggering condition, and evolution of the states for the decentralized event-triggering implementation including adaptation via the vector $\theta$. 34
2.5 Times between updates, evolution of the triggering condition, and evolution of the states for the decentralized event-triggering implementation without adaptation. 35
2.6 Adaptation parameter vector evolution for the adaptive decentralized event-triggered implementation. 35
2.7 $V(\xi(t))$ (solid line) and $V(\xi(t_k))e^{-\lambda(t-t_k)}$ (dotted line) illustrating the triggering of new actuation. 37
2.8 Inter-execution times when no disturbance is present. 37
2.9 Lyapunov function evolution under the presence of a disturbance of norm $\|\delta\|_\infty = 1$. 38
2.10 Inter-execution times under the presence of a disturbance of norm $\|\delta\|_\infty = 1$. 38
2.11 Lyapunov function evolution under the presence of a disturbance of norm $\|\delta\|_\infty = 10$. 39
3.1 DC motor and associated electric circuit. 83

3.2 Simulink diagram for the closed-loop system depicting the symbolic controller block included in Pessoa. 85

3.3 Evolution of velocity and input voltage for the DC motor example. 87

3.4 Evolution of velocity and current when the input voltage is restricted to $-10, 0, \text{ and } 10 \text{ Volts}$. 87

3.5 Evolution of velocity and input when the input voltage is restricted to $-10, 0, \text{ and } 10 \text{ Volts}$. 88

3.6 Evolution of velocity and current when the input voltage is restricted to $-10, 0, \text{ and } 10 \text{ Volts}$ and state constraints are enforced. 89

3.7 Automaton describing the availability of the shared resources. The lower part of the states is labeled with the outputs $a$ and $u$ denoting availability and unavailability of the shared resource, respectively. 89

3.8 Domain of the controller forcing the double integrator to remain in $[-1,1] \times [-1,1]$ under the fairness constraints described by the automaton in Figure 3.7. 91

3.9 Evolution of the state variables (left figure) and inputs (right figure), from initial state $(x_1, x_2) = (-1, 0.8)$, when the automaton in Figure 3.7 is visiting the states $|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|\ldots$. The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta. 92
3.10 Evolution of the state variables (left figure) and inputs (right figure), from initial state \((x_1, x_2) = (-1, 0.8)\), when the automaton in Figure 3.7 is visiting the states \(q_1 q_2 q_1 q_2 q_1 q_2 q_1 q_2 q_1 \ldots\). The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta. ................................................................. 93

3.11 Evolution of the state variables. The left figure refers to the initial states and automaton evolution in Figure 3.9 while the right figure refers to the initial states and automaton evolution in Figure 3.10. ................................................................. 94

3.12 (a) Symbolic controller \(S^*_c\). (b) Time to reach the target set \(W\) represented as the ratio between the times obtained from the symbolic controller and the times obtained from the continuous time-optimal controller to reach the origin. ................................................................. 95

3.13 Unicycle trajectory under the automatically generated approximately time-optimal feedback controller (left figure) and the inputs employed: \(v\) in yellow and \(\omega\) in pink (right figure). ................................................................. 96
LIST OF TABLES

2.1 Parameter values for the system and controller. . . . . . . . . . . 33

3.1 Parameters for the circuit in Figure 3.1 expressed in the interna-
tional system of units. . . . . . . . . . . . . . . . . . . . . . . . . . 84

3.2 Times achieved in simulations by a time-optimal controller to reach
the origin and the symbolic controller. . . . . . . . . . . . . . . . . 94
ACKNOWLEDGMENTS

My most sincere gratitude to all the people that made this thesis possible. First and foremost, to my advisor Paulo Tabuada, for being much more than an academic advisor, for being a mentor and a friend. Next, my gratitude goes to all my family: to my parents for the education that they provided me and all the love and support throughout the years; to my sister for shaping my persona through innumerable fights and adventures together; very specially to my grandmother (“mi yaya”) who contributed to making me be who I am in so many ways, and who would have been so proud to see me graduate; and of course to Anke (my love) for her patience during these sometimes stressful years and all the great adventures and relaxing moments together.

I would also like to thank all my colleagues at UCLA: the “control systems inner-circle” (Adolfo Anta, Amir Emadzadeh and Jitkomut Songsiri) always there to share knowledge and tricks, they made my first year at UCLA so much easier; all my lab mates at the Cyber-Physical System Lab (Adolfo Anta, Anna Davitian, Andre Dias, Mostafa Majidpour, Ramkrishna Pasumarty, Giordano Pola, Elaine Render, Pritam Roy, Soheil Samii, Yusuf Yucel and Majid Zamani) with whom I engaged in so many productive discussions; and, of course, my roommate and friend Darren Creutz for these 5 years of fun co-habitation and all the endless and enlightening discussions, would they be about math, politics or philosophy.

I could not forget to acknowledge all the UCLA community, who have made these past 5 years worth keeping in a special place of my memory. I would like to thank in particular my defense committee, my first academic advisor Prof. P. K. C. Wang, and all the administrative staff, especially Deeona Columbia who helped me solve so many administrative problems. I would also like to thank my advisor at KTH, Prof. Karl H. Johansson, without whom I would have never
reached this goal, and Prof. Emilio Frazzoli at MIT for the opportunity to visit his group and experience the great intellectual environment at LIDS for a few months.

Finally, I would like to thank all the people that I have met in these many years of adventures around the world, because all of you have played an important role on the path that lead me here. Thanks to all my friends from Alcalá de Henares, La Rioja, Groningen, my Erasmus experience in Sweden, Newcastle (Australia) and at MIT, thanks also to the L.A. “night surf” crowd, my adoptive family in San Gabriel, my other adoptive family in Arroyo Grande, Pedro Cifuentes a.k.a Moreno-Bombero (for the visits wherever I go), the Venice Beach crowd (beach volley and friendship) and all other friends in California.
Vita

1979  Born, Logroño, Spain.


2001  Summer Student, CERN, Geneva, Switzerland.


2004-2005  Research Assistant, Polytechnic University of Madrid, Madrid, Spain.

2005  Research and Teaching Assistant, The University of Newcastle, Newcastle, Australia.

2005-2007  Master of Science in Electrical Engineering, University of California, Los Angeles, USA.

2007-2010  Research Assistant, University of California, Los Angeles, USA.

2007-2010  Philosophy Doctor in Electrical Engineering, University of California, Los Angeles, USA.
An ISS self-triggered implementation for linear controllers,
M. Mazo Jr, A. Anta and P. Tabuada.
Automatica, Volume 46, Issue 8, August 2010.

Pessoa: A tool for embedded controller synthesis,
M. Mazo Jr., A. Davitian and P. Tabuada.
22nd International Conference on Computer Aided Verification, CAV2010.

Approximate time-optimal control via approximate alternating simulations,
M. Mazo Jr. and P. Tabuada.

PESSOA: towards the automatic synthesis of correct-by-design control software,

Towards decentralized event-triggered implementations of centralized control laws,
M. Mazo Jr. and P. Tabuada.
CONET, 2010. (CPSWEEK 2010)

Input-to-state stability of self-triggered control systems,
M. Mazo Jr. and P. Tabuada.
On Self-Triggered Control for Linear Systems: Guarantees and Complexity,
M. Mazo Jr., A. Anta and P. Tabuada.
2009 European Control Conference.

On event-triggered and self-triggered control over sensor/actuator networks,
M. Mazo Jr. and P. Tabuada.

Reduction of lateral and longitudinal oscillations of vehicles platooning by means
of decentralized overlapping control,
F. Espinosa, A.M.H. Awawdeh, M. Mazo Jr, J.M. Rodriguez, A. Bocos, M. Man-
zano.

Multi-robot tracking of a moving object using directional sensors,
M. Mazo Jr., A. Speranzon, K. H. Johansson, and X. Hu.

Robust area coverage using hybrid control,
M. Mazo Jr and K. H. Johansson.

Integrated Development Environment for Underactuated Non-Linear Control Sys-
tems,
F. Espinosa, F. J. Castillo, M. Mazo Jr.
Abstract of the Dissertation

Contributions to the Control of Networked Cyber-Physical Systems

by

Manuel Mazo Espinosa

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2010

Professor Paulo Tabuada, Chair

Cyber-Physical Systems (CPSs) are complex engineered systems in which digital computation devices interact with the physical world. Boosted by recent advances in computation, communication, and sensing technologies these systems are becoming increasingly ubiquitous. Cyber-Physical Systems exhibit complex behaviors stemming from intricate interactions between the physical world and computation devices. The increasing presence of CPSs in life critical applications combined with the inability of current methodologies to analyze these systems urges the development of new approaches for analysis and design. Moreover, these new techniques, yet to be developed, are required to deliver CPSs that are both efficient and operate correctly under a wide range of circumstances. This is one of the greatest challenges faced by the cyber-physical systems community. In this thesis I present my contributions to the solution of this problem through two complementary techniques: I provide efficient implementations of decentralized control systems over wireless sensor/actuator networks while guaranteeing operational performance; and I provide automated tools for the synthesis of correct-by-design embedded controllers considering time optimality requirements.
CHAPTER 1

Introduction

1.1 Motivation

In the last decade we have seen great advances, both theoretical and applied, in computation, communication, and control. Electronics development has continued to follow the Moore’s law predictions, which has not only affected the advances in computation but also in communication. Improvements in transducer technologies have provided with smaller and more accurate sensors, and similarly more efficient actuators. Enabled by the progress in miniaturization, these technologies are becoming ubiquitous in our daily life. Complex systems resulting from the combination of these technologies, and in direct interaction with the physical world, have emerged under the name of *Cyber-Physical Systems*. As a result of the intricate interactions between subsystems, and between the physical world and digital information processing devices, these systems exhibit great complexity. This complexity brings new levels of difficulty to the analysis of such systems especially in what regards performance guarantees. Moreover, their increasing presence in applications critical for our daily life imposes strict guarantees of correct operation. These new requirements demand new tools capable of modeling and analyzing systems combining computation, communication, and control. Furthermore, design tools capable of delivering operational guarantees are needed as well. The combination of both efficiency and guaranteed
performance is one of the greatest challenges that modern engineering, and in particular the cyber-physical systems community, faces.

In this thesis I focus on the problem of controlling cyber-physical systems to enforce pre-specified system behaviors. I address two different facets of this problem:

1. on the interface between communication and control, I provide solutions to design efficient implementations of decentralized control systems over wireless sensor/actuator networks, while delivering performance guarantees;

2. on the interface between computation and control, I analyze the use of symbolic models for control design and present a tool for the synthesis of correct-by-design embedded controllers.

In the first part of this thesis several decentralized implementations of control algorithms are proposed. The aforementioned advances in processor, memory, and wireless technologies have enabled the development of small nodes capable of communication, computation, and sensing. Wireless networks of such nodes are already being deployed for distributed sensing applications for which industry already provides commercial products. Moreover, wireless enabled actuators have also started to appear in industrial catalogues. A natural next step is to close the loop over such networks to perform control: while some nodes are used to sense the environment, other nodes are used to alter the physical world based on the collected information. Control applications rely on frequent measurements of the state of the physical system being controlled to update the actuation. Wireless sensing nodes suffer from limited available energy which is normally provided by small batteries. This fact establishes a fundamental limitation as the necessary transmission of measurements from sensing nodes to actuating nodes is
costly in energy. Hence, it is necessary to design control algorithms that achieve desired levels of performance while reducing the frequency of measurements. The controller implementations I introduce in the first part of this thesis provide prescribed levels of performance while reducing the communication requirements.

The second part of the thesis is devoted to the study of symbolic abstractions in control. Symbolic abstractions are simpler descriptions of control systems, typically with finitely many states, in which each symbolic state represents a collection or aggregate of states in the control system. I study the construction of symbolic models for general classes of dynamical systems described by differential equations. Similar models are used in software and hardware modeling, which enables the composition of such models with the symbolic abstraction of the continuous dynamics. The result of this composition are symbolic models capturing the behavior of the complete cyber-physical system. Given specifications, also in the form of finite symbolic models, the synthesis of controllers can be reduced to a fixed-point computation over the (finite-state) symbolic abstraction. The resulting controllers can later be refined into hybrid-controllers, combining continuous and discrete dynamics, that can be deployed on the actual implementation. Following this design flow, the controllers obtained are guaranteed to satisfy the provided specification,justifying the name of correct-by-design synthesis. Moreover, I show that by making use of these symbolic abstractions one can also approximately solve time-optimal control problems. Finally, I present Pessoa, a tool implementing both the construction of symbolic abstractions and the synthesis of correct-by-design controllers.
1.2 Organization of the thesis

This thesis is divided in 4 chapters, the first of which is the current introduction. Chapter 2 is devoted to decentralized implementations of controllers over wireless sensor/actuator networks. Chapter 3 studies the suitability of symbolic models for the synthesis of correct-by-design embedded controllers, and presents a tool developed for this purpose. Finally, the thesis concludes in Chapter 4 with a brief discussion and suggestions for future research.

For clarity of exposition, both Chapter 2 and Chapter 3 follow a common structure. Both chapters start with an introduction including: a description of the problem addressed, a brief literature review, and a statement of the contributions made. Following the introduction, a section establishing the notation and other preliminaries specific to the chapter is included. For the sake of self-containment of these chapters, some notions might be defined twice. The developed techniques are detailed in subsequent sections, followed by a section illustrating their efficiency on simulated examples. The chapters are concluded with a discussion section. For better readability, all the proofs are collected in respective appendixes at the end of these chapters.
CHAPTER 2

Controller implementations over wireless
sensor/actuator networks

2.1 Introduction

Wireless sensor networks have blossomed in the recent years. A large amount of
literature has been devoted to the problem of efficient data collection and dis-
tribution over such networks. The incorporation of wireless actuators in these
networks brought wireless sensor/actuators networks (WSAN) to the forefront of
research. Due to the cheap deployment and the increased versatility of WSAN,
the control community has started to devote attention to such infrastructures
for control applications. However, in these new architectures, control schemes
become more challenging. Power consumption is one of such challenges. Usually,
the sensor nodes are powered autonomously by batteries. The relationship be-
tween the capacity (and hence physical size) of these batteries and the lifespan
of the network will be determined by the energy efficiency of the implemented
algorithms. In this chapter we provide communication-efficient implementations
of decentralized control systems over WSAN. Communication is the most energy
expensive process taking place at the sensor nodes. The implementations we
provide reduce energy consumption by resorting to aperiodic control techniques
which reduce the amount of communication required between sensors, controllers,
and actuators. Moreover, by reducing the number of controller updates, the proposed techniques make an efficient use of the network capacity, which enables other applications to share the same network infrastructure.

The techniques presented here have resulted in the publications [MAT10] and [MT10b], which provide most of the contents of the chapter. Other related results of my research, but not included here, can be found in [MT08].

2.1.1 Previous work

Wireless sensor networks research has extensively dealt with the extraction of information from the physical world. Many of the applications developed concentrate on how to obtain this information for posterior off-line analysis [FHK06, GR06]. Others are concerned with on-line processing of this information for different applications such as tracking [OS05, SSS03, WYE05], distributed optimization [RN04], or mapping [DSG08]. In all of these applications there is a common desire for small power consumption which would extend the life span of the network.

Many of the approaches used to reduce the power consumption concentrate on the communication requirements. Some techniques rely on information theoretic arguments to achieve improvements by compressing the sensed information [BHS06] or efficiently increase the network throughput [CSA04, LHA07]; others, focus on message-passing algorithms, such as Directed Diffusion [IGE03] and Junction Trees [PGM05] or on the sleep-scheduling of the nodes [SF06]. Still, most of these studies are performed under the premise or assumption that the sensor network will only be gathering information for on-line or off-line analysis.

Recently, some work devoted to enabling control applications over wireless networks has started to appear. Control engineers have typically designed their
controllers as if the channels between sensors, controllers, and actuators were infinite-bandwidth, noise-free and delay-free. The effects of non-idealities in the channels, in practice, could be mitigated by employing better hardware. However, on implementations over WSAN these limitations of the communication medium can no longer be neglected. This fact, combined with the recent interest from industry, e.g. the WirelessHART initiative [Wir], has fueled the study of control under communication constraints in the past decade. Much research has been devoted to the effects of: quantization in the sensors; delay and jitter; limited bandwidth; or even packet losses. Some good overviews of these topics can be found in the report resulting from the RUNES project [ABH06], and the special issue of the IEEE proceedings [AB07]. The communications community is also directing efforts to enable reliable wireless networks for control applications. WirelessHART [SHM08] is an example of these efforts to provide wireless communication standards that meet the demands of control applications. Another example is the study of MAC protocols over the existing 802.15.4 hardware (basis for the popular ZigBee [Zig] standard) that could meet low-latency and hard real-time constraints, both desirable in control applications [CV08].

One of the first questions asked when implementing control systems over wireless networks, or any other digital platform, is: how often should one sample the physical environment? Many researchers have worked on the analysis of this sole problem. Tools like the delta-transform [GMP92] were developed, and many books discussed this issue [GGS01, HL84]. More recently, Nesic and collaborators have proposed techniques to select periods retaining closed-loop stability in networked systems [NT01, NTC09]. However, engineers still rely mostly on rules of thumb such as sampling with a frequency 20 times the system bandwidth, and then check if it actually works [Fra07, GGS01, HL84]. A shift in perspective was brought by the notion of event-triggered control [Arz99], [AB02]. In event-
triggered control, instead of periodically updating the control input, the update
instants are generated by the violation of a condition on the state of the plant.
Many researchers have proposed event-triggered implementations in the recent
years [HSB08], [MT08], [WL09a], [Cog09]. In particular, Tabuada proposed a
formalism to generate asymptotically stable event-triggered implementations of
non-linear controllers [Tab07], and in [MT08] the author explored the application
of event-triggered and self-triggered techniques to distributed implementations
of linear controllers. Following the formalism in [Tab07], Wang and Lemmon
proposed a distributed event-triggered implementation for weakly-coupled dis-
tributed systems [WL09c].

The concept of self-triggered control was introduced by Velasco and coworkers
in [VFM03] as another approach for aperiodic control. The key idea of self-
triggered control is to compute, based on the current state measurements, the
next instant of time at which the control law is to be recomputed. In between
updates of the controller the control signal is held constant and the appropriate
generation of the update times guarantees the stability of the closed-loop system.
Under self-triggered implementations the time between updates is a function of
the state, and thus less control executions are expected. On the other hand, the
intervals of time in which no attention is devoted to the plant pose a new concern
regarding the robustness of self-triggered implementations. Several self-triggered
implementations have been proposed in the last years, both for linear [WL09b]
and non-linear [AT10] plants.

The notion of input-to-state stability [Son06] is fundamental in the approach
followed in the present thesis. Finally, the approaches followed in [NT04] and
[KST04] to analyze the effect of external disturbances have greatly influenced
the work hereby presented.
2.1.2 Contributions

In what follows we propose to minimize the energy consumption by resorting to event-triggered and self-triggered sampling strategies over WSAN. We will show how the techniques introduced in [Tab07], and reviewed in Section 2.3, can be implemented over sensor-actuator networks to considerably reduce the number of network transmissions. We propose an event-triggered strategy in which each node uses its local information to determine when to make a transmission and a self-triggered strategy in which the actuator node determines for how long should the sensing nodes sleep before collecting and transmitting fresh measurements.

The first contribution, described in Section 2.4, is a strategy for the construction of decentralized event-triggered implementations over WSAN of centralized controllers. This contribution lead to the publication in [MT10b] from which Section 2.4 has been extracted. The event-triggered techniques introduced in [Tab07] are based on a criterion that depends on the norm of the vector of measured quantities. This is natural in the setting discussed in [Tab07] since sensors were collocated with the micro-controller. However, in a WSAN the physically distributed sensor nodes do not have access to all the measured quantities. Hence, we cannot use the same criterion to determine when the control signal should be re-computed. Using classical observers or estimators (as the Kalman filter) would require filters of dimension as large as the number of states in each sensor node, which would be unpractical given the low computing capabilities of sensor nodes. Moreover, we do not assume observability from every measured output, thus ruling out observer-based techniques. Approaches based on consensus algorithms are also unpractical as they require large amounts of communication and thus large energy expenditures by the sensor nodes. Instead, we present an approach to decentralize a centralized event-triggered condition. Our technique also provides
a mechanism to enlarge the resulting times between controller re-computations without altering performance guarantees.

We do not address in this thesis practical issues such as delays or jitter in the communication and focus solely on the reduction of the actuation frequency (with its associated communication and energy savings). In particular, the issue of communication delays has been shown to be easily addressed in the context of event-triggered control in [Tab07] and similarly in [WL09c]. The approach followed in those papers is applicable to the techniques we present. Moreover, these techniques can be implemented over the WirelessHART standard [Wir], which addresses other communication concerns such as medium access control, power control, and routing. The decentralized implementation that we provide is complementary to the implementation for weakly-coupled distributed systems provided in [WL09c]. We will remark this fact in the discussion at the end of the chapter.

In Section 2.5 we introduce the second contribution in this chapter: a self-triggered implementation for linear systems. That section has been compiled mainly from excerpts of my work published in [MAT10]. In this self-triggered implementation the times between controller updates are as large as possible so as to enforce desired levels of performance subject to the computational limitations of the digital platform. By increasing the available computational resources the performance guarantees improve while the number of controller executions is reduced. Hence, the proposed technique reduces the actuation requirements (and communication, in networked systems) in exchange for computation. Furthermore, we also show that the proposed self-triggered implementation results in an exponentially input-to-state stable closed-loop system with corresponding gains depending on the available computational resources, which addresses
the concerns with respect to robustness to external disturbances of self-triggered implementations. The idea advocated in this part of the chapter, trading communication/actuation for computation, was already explored in [YTS02]. However, their approach is aimed at loosely coupled distributed systems, where local actuation takes place continuously and communication between subsystems is reduced by means of state estimators. Complementary to the work in [AT10] the approach followed in the proposed self-triggered implementation provides large inter-execution times for linear systems by not requiring a continuous decay of the Lyapunov function in use, much in the spirit of [WL08]. Computing exactly the maximum allowable inter-execution times guaranteeing stability requires the solution of transcendental equations for which closed form expressions do not exist. Our proposal computes approximations of these maximum allowable inter-execution times while providing stability guarantees.

2.2 Preliminaries

2.2.1 Notation

We denote by $\mathbb{R}^+$ the positive real numbers. We also use $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$. The usual Euclidean ($l_2$) vector norm is represented by $| \cdot |$. When applied to a matrix $| \cdot |$ denotes the $l_2$ induced matrix norm. A matrix $P \in \mathbb{R}^{m \times m}$ is said to be positive definite, denoted $P > 0$, whenever $x^T P x > 0$ for all $x \neq 0$, $x \in \mathbb{R}^m$, and a matrix $A$ is said to be Hurwitz when all its eigenvalues have strictly negative real part. We denote by $I$ the identity matrix. By $\lambda_m(P), \lambda_M(P)$ we denote the minimum and maximum eigenvalues of $P$ respectively. Given an essentially bounded function $\delta : \mathbb{R}_0^+ \to \mathbb{R}^m$ we denote by $\| \delta \|_\infty$ its $L_\infty$ norm, i.e., $\| \delta \|_\infty = (ess) \sup_{t \in \mathbb{R}_0^+} \{|\delta(t)|\} < \infty$. We denote vectors and vector valued
functions by lower-case letters and denote matrices by upper-case letters. In denoting vector functions we will often drop the explicit dependence on the free variable, i.e. $\xi = \xi(t)$, when there is no confusion or there is no need to remark this time dependence.

2.2.2 Mathematical systems theory

We start by providing a brief review of mathematical systems theory. First we introduce two fundamental definitions:

**Definition 2.2.1 (Lipschitz continuity).** Given an open set $B \subseteq \mathbb{R}^n$, we say that a function $f$ is Lipschitz continuous on $B$ if there exists a constant $L \in \mathbb{R}^+_{0}$ such that:

$$\|f(x) - f(y)\| \leq L\|x - y\|, \forall x, y \in B$$

**Definition 2.2.2 (Control system).** A Control System is a dynamical system described by an ordinary differential equation:

$$\dot{\xi} = f(\xi, \upsilon, \delta), \xi : \mathbb{R}^+_0 \rightarrow \mathbb{R}^n, \upsilon : \mathbb{R}^+_0 \rightarrow \mathbb{R}^m, \delta : \mathbb{R}^+_0 \rightarrow \mathbb{R}^p \quad (2.1)$$

where $\xi$ is known as the state trajectory, $\upsilon$ as the “input” or “control signal” and $\delta$ as the ”disturbance” or ”uncontrolled input”.

Control systems and their solutions are the fundamental objects of study of this thesis. Solutions of a control system with initial condition $x$ and inputs $\upsilon$ and $\delta$, denoted by $\xi_{x\upsilon\delta}(t)$, satisfy: $\xi_{x\upsilon\delta}(0) = x$ and $\frac{d}{dt}\xi_{x\upsilon\delta}(t) = f(\xi_{x\upsilon\delta}, \upsilon(t), \delta(t))$ for almost all $t \in \mathbb{R}^+_0$. The notation will be relaxed by dropping the subindex when it does not contribute to the clarity of exposition. In what follows $f$ will be assumed to be Lipschitz in its arguments so that there always exists a unique solution to the differential equation (2.1).
We will work with two different kinds of control signals: piece-wise constant, and therefore Lipschitz on compacts; or “feedback” control signals, i.e. $v = k(\xi)$ with $k : \mathbb{R}^n \to \mathbb{R}^m$ continuous and differentiable. In this second case $v$ will be continuous and differentiable (as $\xi$ will be the solution to a differential equation), and therefore again Lipschitz. We will refer to the system $\dot{\xi} = f(\xi, k(\xi))$ as the closed loop system. In the present chapter we assume that the closed-loop system is an autonomous system:

**Definition 2.2.3** (Autonomous System). The ordinary differential equation:

$$\dot{\xi} = f(\xi), \; \xi : \mathbb{R}^+_0 \to \mathbb{R}^n, \quad (2.2)$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$ Lipschitz continuous, is said to be autonomous if $f(\xi(t))$ does not depend explicitly on the free variable $t$ (often regarded as time).

In the remainder of the chapter we are mainly concerned with the stability properties of the closed-loop system produced by certain controller implementations. The notion of stability is formalized through the following two definitions:

**Definition 2.2.4** (Equilibrium point). The point $\tilde{x} \in \mathbb{R}^n$ is an equilibrium point for the differential equation (2.2) if $f(\tilde{x}) = 0$.

**Definition 2.2.5** (Stability). Let $\xi(t)$ denote a solution for the differential equation (2.2). The equilibrium point $\tilde{x} \in \mathbb{R}^n$ of (2.2) is said to be:

- **(Lyapunov) Stable** if for every $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that if $\|\xi(0)\| < \delta$ then $\|\xi(t)\| < \epsilon$ for all $t \geq 0$.

- **Asymptotically Stable** if it is stable and there exists $\delta > 0$ such that if $\|\xi(0)\| < \delta$ then $\lim_{t \to \infty} \|\xi(t)\| = \tilde{x}$. 

13
Exponentially Stable if it is asymptotically stable and if there exist constants $M, \beta, \delta > 0$ such that if $\|\xi(0)\| < \delta$ then $\|\xi(t)\| \leq M\|\xi(0)\|e^{-\beta t}$, for all $t \geq 0$

We review now a theorem characterizing the stability of equilibrium points of a dynamical system. For a proof of this theorem we refer the reader to [Kha02, AM06].

**Theorem 2.2.6** (Lyapunov’s Second Theorem on Stability). Let $\tilde{x}$ be an equilibrium point of an ordinary differential equation $\dot{\xi} = f(\xi)$. Consider a function $V : \mathbb{R}^n \to \mathbb{R}$ such that:

- $V(x) > 0$ for all $x \neq \tilde{x}$;
- $V(\tilde{x}) = 0$;
- $\frac{\partial V}{\partial x} f(x) < 0$ for all $x \neq \tilde{x}$;

then the equilibrium point $\tilde{x}$ is asymptotically stable.

A different stability notion playing a fundamental role in the remainder of the thesis is Input-to-State Stability. In order to formalize this notion we need first to introduce two new classes of functions: $\mathcal{K}_\infty$ and $\mathcal{KL}$ functions. A function $\gamma : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, is of class $\mathcal{K}_\infty$ if it is continuous, strictly increasing, $\gamma(0) = 0$ and $\gamma(s) \to \infty$ as $s \to \infty$. A continuous function $\beta : \mathbb{R}_0^+ \times [0, a[ \to \mathbb{R}_0^+$ is of class $\mathcal{KL}$ if $\beta(\cdot, \tau)$ is of class $\mathcal{K}_\infty$ for each $\tau \geq 0$ and $\beta(s, \cdot)$ is monotonically decreasing to zero for each $s \geq 0$. A class $\mathcal{KL}$ function $\beta$ is called exponential if $\beta(s, \tau) \leq \sigma se^{-ct}$, $\sigma > 0, c > 0$.

**Definition 2.2.7** (Input-to-State Stability [Son06]). A control system $\dot{\xi} = f(\xi, v)$ is said to be input-to-state stable (ISS) with respect to $v$ if there exists $\beta \in \mathcal{KL}$
and \( \gamma \in \mathcal{K}_\infty \) such that for any \( t \in \mathbb{R}_0^+ \) and for all \( x \in \mathbb{R}^n \):

\[
|\xi_{xv}(t)| \leq \beta(|x|, t) + \gamma(\|v\|_\infty).
\]

We shall refer to \((\beta, \gamma)\) as the ISS gains of the ISS estimate.

Section 2.5 is devoted to the study of self-triggered implementations for linear time invariant systems.

**Definition 2.2.8** (Linear Time Invariant System). The system defined by the ordinary differential equation (2.2) is said to be linear time invariant if it is autonomous and the right hand side is linear in \( \xi \), i.e. \( \dot{\xi} = A\xi \), for some matrix \( A \in \mathbb{R}^{n \times n} \). For a control system the definition requires linearity of (2.1) on both \( v \) and \( \xi \), i.e. \( \dot{\xi} = A\xi + Bv \), for some matrices \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times m} \).

A linear feedback law for a linear control system is a map \( u = Kx \); we will sometimes refer to such a law as a controller for the system.

When restricted to linear systems the following definitions and stability characterizations are also used:

**Definition 2.2.9** (Lyapunov function). A smooth function \( V : \mathbb{R}^n \to \mathbb{R}_0^+ \) is said to be a Lyapunov function for a linear system \( \dot{\xi} = A\xi \) if there exists class \( \mathcal{K}_\infty \) functions \( \underline{\alpha}, \overline{\alpha} \), and \( \lambda \in \mathbb{R}^+ \) such that for all \( x \in \mathbb{R}^n \):

\[
\underline{\alpha}(|x|) \leq V(x) \leq \overline{\alpha}(|x|)
\]

\[
\frac{\partial V}{\partial x} Ax \leq -\lambda V(x).
\]

We will refer to \( \lambda \) as the rate of decay of the Lyapunov function. In what follows we will consider functions of the form \( V(x) = (x^TPx)^{\frac{1}{2}} \), in which case \( V \) is a Lyapunov function for system \( \dot{\xi} = A\xi \) if and only if \( P > 0 \) and \( A^TP + PA \leq -2\lambda I \) for some \( \lambda \in \mathbb{R}^+ \), the rate of decay.

For a proof of the following theorem we refer the reader again to [AM06]:
Theorem 2.2.10 (Asymptotic Stability of Linear Time Invariant Systems). For the linear time invariant system $\dot{\xi} = A\xi$, the equilibrium point $\bar{x} = 0$ is asymptotically stable if and only if $\text{Re}(\lambda(A)) < 0$.

Definition 2.2.11 (EISS). A control system $\dot{\xi} = A\xi + \delta$ is said to be exponentially input-to-state stable (EISS) if there exists $\lambda \in \mathbb{R}^+$, $\sigma \in \mathbb{R}^+$ and $\gamma \in \mathcal{K}_\infty$ such that for any $t \in \mathbb{R}_0^+$ and for all $x \in \mathbb{R}^n$:

$$|\xi_{xs}(t)| \leq \sigma|x|e^{-\lambda t} + \gamma(\|\delta\|_\infty).$$

We shall refer to $(\beta, \gamma)$, where $\beta(s, t) = s\sigma e^{-\lambda t}$, as the EISS gains of the EISS estimate. If no disturbance is present, i.e., $\delta = 0$, an EISS system is said to be globally exponentially stable (GES).

2.3 Event-triggered control

We begin by revisiting the results from [Tab07], which serve as the basis for the rest of this chapter. Let us start by considering a nonlinear control system:

$$\dot{\xi} = f(\xi, \nu) \quad (2.3)$$

and assume that a feedback control law $k : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $\nu = k(\xi)$ is available, rendering the closed-loop system:

$$\dot{\xi} = f(\xi, k(\xi + \varepsilon)) \quad (2.4)$$

input-to-state stable (ISS) with respect to measurement errors $\varepsilon : \mathbb{R}_0^+ \rightarrow \mathbb{R}^n$. We provide the following characterization of ISS that lies at the heart of our techniques:

Definition 2.3.1. A smooth function $V : \mathbb{R}^n \rightarrow \mathbb{R}_0^+$ is said to be an ISS Lyapunov function for the closed-loop system (2.4) if there exists class $\mathcal{K}_\infty$ functions $\underline{\alpha}, \overline{\alpha}$,
α and γ such that for all \( x \in \mathbb{R}^n \) and \( e \in \mathbb{R}^n \) the following is satisfied:

\[
\alpha(|x|) \leq V(x) \leq \overline{\alpha}(|x|)
\]

\[
\frac{\partial V}{\partial x} f(x, k(x + e)) \leq -\alpha(|x|) + \gamma(|e|). \tag{2.5}
\]

The closed-loop system (2.4) is said to be ISS with respect to measurement errors \( \varepsilon \), if there exists an ISS Lyapunov function for (2.4).

In a sample-and-hold implementation of the control law \( v = k(\xi) \), the input signal is held constant between update times, i.e.:

\[
\begin{align*}
\dot{\xi}(t) &= f(\xi(t), v(t)) \\
v(t) &= k(\xi(t_k)), 
&\quad t \in [t_k, t_{k+1}]. \tag{2.6}
\end{align*}
\]

where \( \{t_k\}_{k \in \mathbb{N}_0^+} \) is a divergent sequence of update times. An event-triggered implementation defines such a sequence of update times \( \{t_k\}_{k \in \mathbb{N}_0^+} \) for the controller, rendering the closed loop system asymptotically stable.

We now consider the signal \( \varepsilon : \mathbb{R}_0^+ \rightarrow \mathbb{R}^n \) defined by \( \varepsilon(t) = \xi(t_k) - \xi(t) \) for \( t \in [t_k, t_{k+1}] \) and regard it as a measurement error. By doing so, we can rewrite (2.12) for \( t \in [t_k, t_{k+1}] \) as:

\[
\begin{align*}
\dot{\xi}(t) &= f(\xi(t), k(\xi(t) + \varepsilon(t))), \\
\dot{\varepsilon}(t) &= -f(\xi(t), k(\xi(t) + \varepsilon(t))), 
&\quad \varepsilon(t_k) = 0.
\end{align*}
\]

Hence, as (2.4) is ISS with respect to measurement errors \( \varepsilon \), from (2.5) we know that by enforcing:

\[
\gamma(|\varepsilon(t)|) \leq \rho \alpha(|\xi(t)|), \quad \forall t > 0, \quad \rho \in ]0, 1[ \tag{2.7}
\]

the following holds:

\[
\frac{\partial V}{\partial x} f(x, k(x + e)) \leq -(1 - \rho)\alpha(|x|), \quad \forall x, e \in \mathbb{R}^n
\]
and asymptotic stability of the closed-loop follows. Moreover, if one assumes that the system operates in some compact set \( S \subseteq \mathbb{R}^n \) and \( \alpha^{-1} \) and \( \gamma \) are Lipschitz continuous on \( S \), the inequality (2.7) can be replaced by the simpler inequality 
\[
|\varepsilon(t)|^2 \leq \sigma |\xi(t)|^2,
\]
for a suitably chosen \( \sigma > 0 \). Hence, if the sequence of update times \( \{t_k\}_{k \in \mathbb{N}_0^+} \) is such that:
\[
|\varepsilon(t)|^2 \leq \sigma |\xi(t)|^2, \quad t \in [t_k, t_{k+1}[, \quad (2.8)
\]
the sample-and-hold implementation (2.12) is guaranteed to render the closed loop system asymptotically stable.

Condition (2.8) defines an event-triggered implementation that consists of continuously checking (2.8) and triggering the recomputation of the control law as soon as the inequality evaluates to equality. Note that recomputing the controller at time \( t = t_k \) requires a new state measurement and thus resets the error \( \varepsilon(t_k) = \xi(t_k) - \xi(t_k) \) to zero which enforces (2.8).

### 2.4 Decentralized event-triggered control

In this section we discuss an implementation of a control system in event-triggered form over sensor/actuator networks. We consider scenarios in which three kinds of nodes are present in the network: sensing, computing and actuation nodes. In this work we consider the case in which just one computing node is present. We also assume, for simplicity of presentation, a decentralized scenario in which each state is measured by a different sensor. However, the same ideas apply to more general decentralized scenarios as we briefly discuss at the end of this section. Such a scenario is presented in figure 2.1 where blue circles denote sensing nodes, red diamonds denote actuation nodes, and the green square is the computing node. This is in fact a typical configuration considered in WirelessHART [ZSJ09].
In this setting, no sensor can evaluate condition (2.8), since (2.8) requires the knowledge of the full state vector $\xi(t)$. Our goal is to provide a set of simple conditions that each sensor can check locally to decide when to trigger a controller update, thus triggering also the transmission of fresh measurements from sensors to the controller.

### 2.4.1 Decentralized conditions

Using a set of parameters $\theta_1, \theta_2, \ldots, \theta_n \in \mathbb{R}$ such that $\sum_{i=1}^{n} \theta_i = 0$, we can rewrite inequality (2.8) as:

$$\sum_{i=1}^{n} \varepsilon^2_i(t) - \sigma \xi^2_i(t) \leq 0 = \sum_{i=1}^{n} \theta_i,$$

where $\varepsilon_i$ and $\xi_i$ denote the $i$-th coordinates of $\varepsilon$ and $\xi$ respectively. Hence, the following implication holds:

$$\bigwedge_{i=1}^{n} (\varepsilon^2_i(t) - \sigma \xi^2_i(t) \leq \theta_i) \Rightarrow |\varepsilon(t)|^2 \leq \sigma |\xi(t)|^2, \quad (2.9)$$

which suggests the use of:

$$\varepsilon^2_i(t) - \sigma \xi^2_i(t) \leq \theta_i \quad (2.10)$$

as the local event-triggering conditions.
In this decentralized scheme, whenever any of the local conditions (2.10) becomes an equality, the controller is recomputed. We denote by $t_k + \tau_i(x)$ the first time at which (2.10) is violated, when $\xi(t_k) = x$, $\varepsilon(t_k) = 0$. If the time elapsed between two events triggering controller updates is smaller than the minimum time $\tau_{min}$ between updates of the centralized event-triggered implementation\(^1\), the second event is ignored and the controller update is scheduled $\tau_{min}$ units of time after the previous update.

Not having an equivalence in (2.9) entails that this decentralization approach is in general conservative: times between updates will be shorter than in the centralized case. The vector of parameters $\theta = [\theta_1 \theta_2 \ldots \theta_n]^T$ can be used to reduce the mentioned conservatism and thus reduce utilization of the communication network. It is important to note that the vector $\theta$ can change every time the control input is updated. From here on we show explicitly this time dependence of $\theta$ by writing $\theta(k)$ to denote its value between the update instants $t_k$ and $t_{k+1}$. Following the presented approach, as long as $\theta$ satisfies $\sum_{i=1}^{n} \theta_i(k) = 0$, the stability of the closed-loop is guaranteed regardless of the specific value that $\theta$ takes and the rules used to update $\theta$.

We summarize the previous discussion in the following proposition:

**Proposition 2.4.1.** For any choice of $\theta$ satisfying:

$$\sum_{i=1}^{n} \theta_i(k) = 0, \forall k \in \mathbb{N}_0^+,$$

the sequence of update times $\{t_k\}_{k \in \mathbb{N}_0^+}$ given by:

$$t_{k+1} = t_k + \max\{\tau_{min}, \min\{\tau_1(\xi(t_k)), \tau_2(\xi(t_k)), \ldots, \tau_n(\xi(t_k))\}\}$$

renders the system (2.12) asymptotically stable.

\(^1\)It was proved in \cite{Tab07} that such a minimum time exists for the centralized condition, and that lower bounds can be explicitly computed.
2.4.2 Adaptation

We present now a family of heuristics to adjust the vector $\theta$ whenever the control input is updated. We define the decision gap at sensor $i$ at time $t \in [t_k, t_{k+1}]$ as:

$$G_i(t) = \varepsilon_i^2(t) - \sigma \xi_i^2(t) - \theta_i(k).$$

The heuristic aims at equalizing the decision gap at some future time. We propose a family of heuristics parametrized by an equalization time $t_e$ and an approximation order $q$.

For the equalization time $t_e : \mathbb{N}_0 \to \mathbb{R}^+$ we present the following two choices: constant and equal to the minimum time between controller updates $t_e(k) = \tau_{\text{min}}$; the previous time between updates $t_e(k) = t_k - t_{k-1}$.

The approximation order is the order of the Taylor expansion used to estimate the decision gap at the equalization time $t_e$:

$$\hat{G}_i(t_k + t_e) = \hat{\varepsilon}_i^2(t_k + t_e) - \sigma \hat{\xi}_i^2(t_k + t_e) - \theta_i(k).$$

where for $t \in [t_k, t_{k+1}]$:

$$\hat{\xi}_i(t) = \xi_i(t_k) + \dot{\xi}_i(t_k)(t - t_k) + \frac{1}{2} \ddot{\xi}_i(t_k)(t - t_k)^2 + \ldots + \frac{1}{q!} \xi_i^{(q)}(t_k)(t - t_k)^q,$$

$$\hat{\varepsilon}_i(t) = 0 - \dot{\xi}_i(t_k)(t - t_k) - \frac{1}{2} \ddot{\xi}_i(t_k)(t - t_k)^2 - \ldots - \frac{1}{q!} \xi_i^{(q)}(t_k)(t - t_k)^q,$$

using the fact that $\dot{\varepsilon} = -\hat{\xi}$ and $\varepsilon(t_k) = 0$.

Finally, once an equalization time $t_e$ and an approximation order $q$ are chosen, the vector $\theta(k) \in \mathbb{R}^n$ is computed so as to satisfy:

$$\forall i, j \in \{1, 2, \ldots, n\}, \quad \hat{G}_i(t_k + t_e) = \hat{G}_j(t_k + t_e), \quad \sum_{i=1}^n \theta_i(k) = 0.$$

Note that finding such $\theta$, after the estimates $\hat{\xi}$ and $\hat{\varepsilon}$ have been computed, amounts to solving the system of $n$ linear equations:
\[
\begin{bmatrix}
1 & -1 & 0 & 0 & \ldots & 0 \\
0 & 1 & -1 & 0 & \ldots & 0 \\
0 & 0 & \ddots & \ddots & \ddots & 0 \\
0 & 0 & 0 & \ldots & 1 & -1 \\
1 & 1 & 1 & \ldots & 1 & 1
\end{bmatrix}
\begin{bmatrix}
\theta_1(k) \\
\theta_2(k) \\
:\ \\
\theta_{n-1}(k) \\
\theta_n(k)
\end{bmatrix}
= \begin{bmatrix}
\delta_{12}(t_k + t_e) \\
\delta_{23}(t_k + t_e) \\
:\ \\
\delta_{(n-2)(n-1)}(t_k + t_e) \\
\delta_{(n-1)n}(t_k + t_e)
\end{bmatrix}
\] (2.11)

\[\delta_{ij}(t) = \left(\dot{\xi}_{i}^{2}(t) - \sigma \dot{\xi}_{i}^{2}(t)\right) - \left(\dot{\xi}_{j}^{2}(t) - \sigma \dot{\xi}_{j}^{2}(t)\right).\]

Note also that \(\theta\) is computed in the controller node, which has access to \(\xi(t_k)\).

The resulting \(\theta\) computed in this way could be such that for some sensor \(i\), \(-\dot{\xi}_{i}^{2}(t_k) > \theta_i(k)\). Such choice of \(\theta\) results in an immediate violation of the triggering condition at \(t = t_k\), i.e., \(\tau_i(\xi(t_k))\) would be zero. In practice, when the unique solution of (2.11) results in \(-\dot{\xi}_{i}^{2}(t_k) > \theta_i(k)\), one resets \(\theta\) to some default value such as the zero vector.

The choice of \(t_e\) and \(q\) has a great impact on the amount of actuation required. The use of a large \(t_e\) leads, in general, to poor estimates of the state of the plant at time \(t_k + t_e\) and thus degrades the equalization of the gaps. On the other hand, one expects that equalizing at times \(t_k + t_e\) as close as possible to the next update time \(t_{k+1}\) (according to the centralized event-triggered implementation) provides larger times between updates. In practice, these two objectives (small \(t_e\), and \(t_{k+1} + t_e\) close to the ideal \(t_{k+1}\)) can be contradictory, namely when the time between controller updates is large. The effect of the order of approximation \(q\) depends heavily on \(t_e\) and enlarging \(q\) does not necessarily improve the estimates.

An heuristic providing good results in several case studies performed by the author is given by Algorithm 1.

While we assumed, for simplicity of presentation, that each node measured a
Input: \( q, t_{k-1}, t_k, \tau_{\text{min}}, \xi(t_k) \)

Output: \( \theta(k) \)

\( t_e := t_k - t_{k-1}; \)

Compute \( \theta(k) \) according to equation (2.11);

if \( \exists i \in \{1,2,\ldots,n\} \) such that \( -\xi_i^2(t_k) > \theta_i(k) \) then

\( t_e := \tau_{\text{min}}; \)

Compute \( \theta(k) \) according to equation (2.11);

if \( \exists i \in \{1,2,\ldots,n\} \) such that \( -\xi_i^2(t_k) > \theta_i(k) \) then

\( \theta(k) := 0; \)

end

end

\textbf{Algorithm 1:} The \( \theta \)-adaptation heuristic algorithm.

single state of the system, in practice there maybe scenarios in which one sensor has access to several (but not all) states of the plant. The same approach applies by considering local triggering rules of the kind \(|\bar{\varepsilon}_i(t)|^2 - \sigma|\bar{\xi}_i(t)|^2 \leq \theta_i\), where \( \bar{\xi}_i(t) \) is now the vector of states sensed at node \( i \), \( \bar{\varepsilon}_i(t) \) is its corresponding error vector, and \( \theta_i \) is a scalar.

2.5 Self-triggered control for linear systems

We take a shift of perspective now, and consider self-triggered implementations of controllers. As briefly described in the introduction, in a self-triggered implementation the controller decides based on the most recently acquired measurements when the next controller update should take place. In a networked setting the application of these techniques can result in great savings on the number of transmissions between sensors, controller, and actuators. The main concern in this kind of implementations is their response to external disturbances, as, in
between controller updates, the controller does not devote any attention to the plant under control. We start by providing a formal definition of the problem we solve.

Consider the sampled-data system:

\[
\begin{align*}
\dot{\xi}(t) &= A\xi(t) + Bv(t) + \delta(t) \\
v(t) &= K\xi(t_k), \quad t \in [t_k, t_{k+1}]
\end{align*}
\] (2.12) (2.13)

where \(\{t_k\}_{k \in \mathbb{N}}\) is a divergent sequence of update times for the controller, and \(A + BK\) is Hurwitz. The signal \(\delta\) can be used to describe measurement disturbances, actuation disturbances, unmodeled dynamics, or other sources of uncertainty as described in [KST04].

A self-triggered implementation of the linear stabilizing controller (2.13) for the plant (2.12) is given by a map \(\Gamma : \mathbb{R}^m \rightarrow \mathbb{R}^+\) determining the controller update time \(t_{k+1}\) as a function of the state \(\xi(t_k)\) at the time \(t_k\), i.e., \(t_{k+1} = t_k + \Gamma(\xi(t_k))\). If we denote by \(\tau_k\) the inter-execution times \(\tau_k = t_{k+1} - t_k\), we have \(\tau_k = \Gamma(\xi(t_k))\). Once the map \(\Gamma\) is defined, the expression \(\text{closed-loop system}\) refers to the sampled-data system (2.12) and (2.13) with the update times \(t_k\) defined by \(t_{k+1} = t_k + \Gamma(\xi(t_k))\).

Formally, the problem we solve in this section is the following:

**Problem 2.5.1.** Given a linear system (2.12) and a linear stabilizing controller (2.13), construct a self-triggered implementation \(\Gamma : \mathbb{R}^m \rightarrow \mathbb{R}^+\) of (2.13) that renders EISS the closed-loop system defined by (2.12), (2.13), while enlarging the inter-execution times.

Having EISS guarantees addresses the problem of robustness to external disturbances, and allows designers to take into account worst-case scenarios in their designs.
2.5.1 Implementation

In order to formally define the self-triggered implementation proposed, we need to introduce two maps:

- $h_c$, a continuous-time output map and
- $h_d$, a discrete-time version of $h_c$.

Let $V$ be a Lyapunov function of the form $V(x) = (x^T P x)^{\frac{1}{2}}$ for $\dot{x} = (A + BK) x$, with rate of decay $\lambda_0$. The output map $h_c : \mathbb{R}^m \times \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$ is defined by:

$$h_c(x, t) := V(\xi_x(t)) - V(x) e^{-\lambda t}$$

for some $0 < \lambda < \lambda_0$. Note that by enforcing:

$$h_c(\xi_x(t_k), t) \leq 0, \quad \forall t \in [0, \tau_k] \quad \forall k \in \mathbb{N}$$

the closed-loop system satisfies:

$$V(\xi_x(t)) \leq V(x) e^{-\lambda t}, \quad \forall t \in \mathbb{R}_0^+ \quad \forall x \in \mathbb{R}^m$$

which implies exponential stability of the closed-loop system in the absence of disturbances, i.e., when $\delta(t) = 0$ for all $t \in \mathbb{R}_0^+$.

Our objective is to construct a self-triggered implementation enforcing (2.15). Since no digital implementation can check (2.15) for all $t \in [t_k, t_{k+1}]$, we consider instead the following discrete-time version of (2.15) based on a sampling time $\Delta \in \mathbb{R}_0^+$:

$$h_d(\xi_x(t_k), n) := h_c(\xi_x(t_k), n\Delta) \leq 0 \quad \forall n \in \left[0, \left\lfloor \frac{\tau_k}{\Delta} \right\rfloor \right]$$

and for all $k \in \mathbb{N}$. This results in the following self-triggered implementation where we use $N_{\min} := \left\lfloor \tau_{\min}/\Delta \right\rfloor$, $N_{\max} := \left\lceil \tau_{\max}/\Delta \right\rceil$, and $\tau_{\min}$ and $\tau_{\max}$ are design parameters. A similar approach was followed in [HSB08] in the context of event-triggered control.
Definition 2.5.2. The map $\Gamma_d : \mathbb{R}^n \rightarrow \mathbb{R}^+$ defined by:

$$\Gamma_d(x) := \max\{\tau_{\min}, n_k \Delta\}$$

$$n_k := \max_{n \in \mathbb{N}}\{n \leq N_{\max}|h_d(x, s) \leq 0, s = 0, \ldots, n\}$$

prescribes a self-triggered implementation of the linear stabilizing controller (2.13) for plant (2.12).

Note that the role of $\tau_{\min}$ and $\tau_{\max}$ is to enforce explicit lower and upper bounds, respectively, for the inter-execution times of the controller. The upper bound enforces robustness of the implementation and limits the computational complexity.

We provide bounds on the computational complexity in the following discussion. The map $h_d(\xi(t_k), n)$ employed to define the implemented triggering condition can be rewritten as $T(n)\nu_2(\xi(t_k))$, where

$$\nu_2(x) = [x_1^2 \ x_1x_2 \ \ldots \ x_1x_m \ x_2^2 \ x_2x_3 \ \ldots \ x_2x_m]^T \in \mathbb{R}^{m(m+1)/2}$$

is the Veronese map of order 2, and $x_i$ denotes the i-th component of the state vector $x$. Using the representation $T(n)\nu_2(\xi(t_k))$ reduces the number of computations and amount of memory necessary for the implementation. Assuming equal complexity for addition, product and comparison operations, the complexity of this implementation is summarized in the following theorem:

Theorem 2.5.3 (Implementation Complexity). The self-triggered implementation for linear control systems summarized in Definition 2.5.2 requires $M_s$ units of memory and $M_t$ computations per controller execution, where:

$$M_s := q \frac{m(m+1)}{2}$$

$$M_t := 2M_s + q + \frac{m(m+1)}{2}$$

$$q := N_{\max} - N_{\min}$$
2.5.2 Input to state stability

In the results presented below, the following functions will be used to define EISS-gains:

\[
\rho_P := \left( \frac{\lambda_M(P)}{\lambda_m(P)} \right)^{\frac{1}{2}}, \quad \gamma_{P,T}(s) := s \frac{\lambda_M(P)}{\lambda_m(P)} \int_0^T |e^{Ar}| dr.
\]

We start by establishing a result explaining how the design parameter \( \tau_{\text{min}} \) should be chosen. The function \( \Gamma_d \) can be seen as a discrete-time version of the function \( \Gamma_c : \mathbb{R}^m \to \mathbb{R}_0^+ \) defined by:

\[
\Gamma_c(x) := \max_{\tau \in \mathbb{R}_0^+} \{ \tau \leq \tau_{\text{max}} | h_c(x,s) \leq 0, \forall s \in [0, \tau] \}. \tag{2.16}
\]

If we use \( \Gamma_c \) to define an ideal self-triggered implementation, the resulting inter-execution times are no smaller than \( \tau^*_\text{min} \) which can be computed as detailed in the next result.

**Lemma 2.5.4.** The inter-execution times generated by the self-triggered implementation in (2.16) are lower bounded by:

\[
\tau^*_\text{min} = \min \{ \tau \in \mathbb{R}^+ : \det M(\tau) = 0 \} \tag{2.17}
\]

where:

\[
M(\tau) := C(e^{FT^*}C^TPC e^{FT^*} - C^TPC e^{-\lambda \tau})C^T, \quad F := \begin{bmatrix} A + BK & BK \\ -A - BK & -BK \end{bmatrix}, \quad C := [I \ 0].
\]

We remind the reader that the proofs of all the results can be found in the Appendix at the end of this chapter. The computation of \( \tau^*_\text{min} \) described in Lemma 2.5.4 can be regarded as a formal procedure to find a sampling period for periodic implementations (also known as maximum allowable time interval or MATI). It should be contrasted with the frequently used ad-hoc rules of
thumb [AW90], [HLA05] (which do not provide stability guarantees). Moreover, an analysis similar to the one in the proof of this lemma can also be applied, *mutatis mutandis*, to other Lyapunov-based triggering conditions, like the ones appearing in [Tab07] and [WL09b]. Notice that the self-triggered approach always renders times no smaller than the periodic implementation, since under a periodic implementation the controller needs to be executed every $\tau_{\text{min}}^*$ (in order to guarantee performance under all possible operating points).

The second and main result establishes EISS of the proposed self-triggered implementation and details how the design parameters $\tau_{\text{min}}, \tau_{\text{max}}, \Delta,$ and $\lambda$ affect the EISS-gains.

**Theorem 2.5.5.** *If $\tau_{\text{min}} \leq \tau_{\text{min}}^*$, the self-triggered implementation in Definition 2.5.2 renders the closed-loop system EISS with gains $(\beta, \gamma)$ given by:

$$
\begin{align*}
\beta(s, t) & := \rho g(\Delta, N_{\text{max}}) e^{-M s}, \\
\gamma(s) & := \gamma_{P, N_{\text{max}}} \Delta(s) \left( \frac{\lambda_{m}^{-\frac{1}{2}}(P)e^{-\lambda_{m} s}}{1 - e^{-\lambda_{m} s}} \right) + \gamma_{I, N_{\text{max}}}(s)
\end{align*}
$$

where:

$$
\begin{align*}
g(\Delta, N_{\text{max}}) & := \rho \left( e^{\frac{(\rho + 2\lambda)\mu\Delta - \rho}{\rho - \rho}} + e^{2\lambda (N_{\text{max}} - 1) \Delta} \left( e^{\frac{(\rho + 2\lambda - \rho)}{\rho - \rho}} - e^{2\lambda m \Delta} \right) \right)^{\frac{1}{2}}, \\
\rho & := \lambda_{M}(G), \quad \mu := \lambda_{m}(G), \\
G & := \begin{bmatrix}
P_{\frac{1}{2}}^{\frac{1}{2}} A P^{-\frac{1}{2}} + (P_{\frac{1}{2}}^{\frac{1}{2}} A P^{-\frac{1}{2}})^{T} & P_{\frac{1}{2}}^{\frac{1}{2}} B K P^{-\frac{1}{2}} \\
(P_{\frac{1}{2}}^{\frac{1}{2}} B K P^{-\frac{1}{2}})^{T} & 0
\end{bmatrix}.
\end{align*}
$$

Note that while $\tau_{\text{min}}$ is constrained by $\tau_{\text{min}}^*$, $\tau_{\text{max}}$ can be freely chosen. However, by enlarging $\tau_{\text{max}}$ (and thus $N_{\text{max}}$) we are degrading the EISS-gains. It is also worth noting that by enlarging $\tau_{\text{max}}$ one can allow longer inter-execution times, and compensate the performance loss by decreasing $\Delta$, at the cost of performing more computations.*
Let us define the maximum exact inter-execution time from $x$ as:

$$\tau^*(x) := \min\{\Gamma_c(x), \tau_{\text{max}}\},$$

where the upper bound is required to obtain robustness against disturbances. The third and final result states that the proposed self-triggered implementation is optimal in the sense that it generates the longest possible inter-execution times given enough computational resources. Hence, by enlarging the inter-execution times we are effectively trading actuation for computation. The proof of the following proposition follows from the proof of Theorem 2.5.5.

**Proposition 2.5.6.** The inter-execution times provided by the self-triggered implementation in Definition 2.5.2 are bounded from below as follows:

$$\Gamma_d(x) \geq \tau^*(x) - \Delta, \ \forall x \in \mathbb{R}^m.$$ 

Note that even if $\Gamma_d(x) \geq \tau^*(x)$ the performance guarantees provided in Theorem 2.5.5 still hold.

**Remark 2.5.7.** When implementing self-triggered policies on digital platforms several issues related to real-time scheduling need to be addressed. For a discussion of some of these issues we refer the readers to [AT09]. Here, we describe the minimal computational requirements for the proposed self-triggered implementation under the absence of other tasks. There are three main sources of delays: measurement, computation, and actuation. Since the computation delays dominate the measurement and actuation delays, we focus on the former. The computation of $\Gamma_d$ is divided in two steps: a preprocessing step performed once by execution, and a running step performed $n$ times when computing $h_d(x,n)$. The preprocessing step has time complexity $(m^2 + m)/2$ and the running step has time complexity $m^2 + m$. If we denote by $\tau_c$ the time it takes to execute an instruction
in a given digital platform, the self-triggered implementation can be executed if:

\[ \frac{3}{2}(m^2 + m)\tau_c \leq \tau_{\text{min}}, \quad (m^2 + m)\tau_c \leq \Delta. \]

The first inequality imposes a minimum processing speed for the digital platform while the second equality establishes a lower bound for the choice of \( \Delta \).

### 2.6 Examples

#### 2.6.1 Decentralized event-triggered control

We present in what follows an example illustrating the effectiveness of the proposed decentralized event-triggered implementation with adaptation. We select the quadruple-tank model from [JA07] describing the multi-input multi-output nonlinear system consisting of four water tanks as shown in Figure 2.2. The water flows from tanks 3 and 4 into tanks 1 and 2, respectively, and from these two tanks to a reservoir. The state of the plant is composed of the water levels of the tanks: \( \xi_1, \xi_2, \xi_3 \) and \( \xi_4 \). Two inputs are available: \( v_1 \) and \( v_2 \), the input flows to the tanks. The input flows are split at two valves \( \gamma_1 \) and \( \gamma_2 \) into the four tanks. The positions of these valves are given as parameters of the plant. The goal is to stabilize the levels \( x_1 \) and \( x_2 \) of the lower tanks at some specified values \( x_1^* \) and \( x_2^* \).

The system dynamics are given by the equation:

\[ \dot{\xi}(t) = f(\xi(t)) + g_c v, \]
with:

\[
\begin{bmatrix}
- \frac{a_1 \sqrt{2g x_1}}{A_1} + \frac{a_3 \sqrt{2g x_3}}{A_1} \\
- \frac{a_2 \sqrt{2g x_2}}{A_2} + \frac{a_4 \sqrt{2g x_4}}{A_2} \\
- \frac{a_3 \sqrt{2g x_3}}{A_3} \\
- \frac{a_4 \sqrt{2g x_4}}{A_4}
\end{bmatrix},
\]

\[
g_c = \begin{bmatrix}
\frac{\gamma_1}{A_1} & 0 \\
0 & \frac{\gamma_2}{A_2}
\end{bmatrix},
\]

and \( g \) denoting gravity’s acceleration and \( A_i \) and \( a_i \) denoting the cross sections of the \( i-th \) tank and outlet hole respectively.

The controller design from [JA07] requires the extension of the plant with two extra artificial states \( \xi_5 \) and \( \xi_6 \). These states are non-linear integrators used by the controller to achieve zero steady-state offset and evolve according to:

\[
\dot{\xi}_5(t) = k_{I_1} a_1 \sqrt{2g} \left( \sqrt{\xi_1(t)} - \sqrt{x_1^*} \right),
\]

\[
\dot{\xi}_6(t) = k_{I_2} a_2 \sqrt{2g} \left( \sqrt{\xi_2(t)} - \sqrt{x_2^*} \right),
\]

where \( k_{I_1} \) and \( k_{I_2} \) are design parameters of the controller. Note how stabilizing the extended system implies that in steady-state \( \xi_1 \) and \( \xi_2 \) converge to the desired values \( x_1^* \) and \( x_2^* \). We assume in our implementation that the sensors measuring \( \xi_1 \) and \( \xi_2 \), also compute \( \xi_5 \) and \( \xi_6 \) respectively.

The controller proposed in [JA07] is given by the following feedback law:

\[
v(t) = -K(\xi(t) - x^*) + u^* \quad (2.18)
\]
with
\[ u^* = \begin{bmatrix} \gamma_1 & 1 - \gamma_2 \\ 1 - \gamma_1 & \gamma_2 \end{bmatrix}^{-1} \begin{bmatrix} a_1 \sqrt{2g} x^*_1 \\ a_2 \sqrt{2g} x^*_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 - \gamma_2 \\ 1 - \gamma_1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} a_1 \sqrt{2g} x^*_3 \\ a_2 \sqrt{2g} x^*_4 \end{bmatrix}, \]
(2.19)
and \( K = QP \) where \( Q \) is a positive definite matrix and \( P \) is given by
\[ P = \begin{bmatrix} \gamma_1 k_1 & (1 - \gamma_1) k_2 & 0 & (1 - \gamma_1) k_4 & \gamma_1 k_1 & (1 - \gamma_1) k_2 \\ (1 - \gamma_2) k_1 & \gamma_2 k_2 & (1 - \gamma_2) k_3 & 0 & (1 - \gamma_2) k_1 & \gamma_2 k_2 \end{bmatrix}, \]
where \( k_1, k_2, k_3 \) and \( k_4 \) are design parameters of the controller. Note how equation (2.19) provides a mean to compute \( x^*_3 \) and \( x^*_4 \) from the specified \( x^*_1 \) and \( x^*_2 \). When computing the control \( \upsilon \), the remaining entries \( x^*_5 \) and \( x^*_6 \) of \( x^* = [x^*_1 \ x^*_2 \ x^*_3 \ x^*_4 \ x^*_5 \ x^*_6]^T \) can be set to any arbitrary (fixed) values \( \hat{x}^*_5 \) and \( \hat{x}^*_6 \). This can be done because the errors: \( \hat{x}^*_5 - x^*_5 \) and \( \hat{x}^*_6 - x^*_6 \), between the arbitrary values and the actual states \( x^*_5 \) and \( x^*_6 \) of the equilibrium, can be reinterpreted as a perturbation on the initial states \( \xi^*_5(0) \) and \( \xi^*_6(0) \).

Using this controller the following function:
\[ H_d(x) = \frac{1}{2} (x - x^*)^T P^T QP (x - x^*) - u^*^T Px + \sum_{i=1}^{4} \frac{2}{3} k_i a_i x_i^{3/2} \sqrt{2g} + k_1 a_1 x_5 \sqrt{2g x_1^*} + k_2 a_2 x_6 \sqrt{2g x_2^*}, \]
(2.20)
which is positive definite and has a global minimum at \( x^* \), is an ISS Lyapunov function with respect to \( \varepsilon \), as evidenced by the following bound:
\[ \frac{d}{dt} H_d(\xi) \leq -\lambda_m(R) |\nabla H_d(\xi)|^2 + |\nabla H_d(\xi)||g_c K||\varepsilon|. \]

This equation suggests the use of the triggering condition:
\[ |\nabla H_d(\xi)||g_c K||\varepsilon| \leq (1 - \rho) \lambda_m(R) |\nabla H_d(\xi)|^2, \rho \in ]0, 1[. \]
Moreover, assuming the operation of the system to be confined to a compact set containing a neighborhood of \( x^* \), the quantity \( |\nabla H_d(\xi)| \) can be bounded as
\[ |\nabla H_d(\xi)| \geq \rho_m |\xi - x^*| \]
and the following triggering rule can be applied to ensure asymptotic stability:

\[ |\varepsilon(t)|^2 \leq \sigma |\xi(t) - x^*|^2, \quad \sigma = \rho_m (1 - \rho) \frac{\lambda_m(R)}{|g_cK|} > 0. \]

We simulated the decentralized event-triggered implementation of this controller following the techniques in Section 2.4. The physical parameters of the plant and the parameters of the controller have been taken from the implementation in [JA07], and are summarized in Table 2.1. Assuming that the system operates in the compact defined by

\[ S = \left\{ x \in \mathbb{R}^6 \mid 0 \leq x_i \leq 20, \ i = 1, \ldots, 6 \right\} \]

and for the choice of \( \rho = 0.75 \), a value of \( \sigma = 0.0038 \) was selected. A bound for the minimum time between controller updates, computed as explained in [Tab07], is given by \( \tau_{\text{min}} = 0.0033 \text{s} \). The decentralized event-triggered controller is implemented adapting \( \theta \) as specified by Algorithm 1 with \( q = 1 \). Furthermore, the pairs of states \( x_1, x_5 \) and \( x_2, x_6 \) are assumed to be measured at the same sensor node, and therefore combined in a single triggering condition at the respective nodes. For comparison purposes, we present in Figure 2.3 the time between controller updates, the evolution of the ratio \( \varepsilon/\xi \) vs \( \sigma \) and the state trajectories, for a centralized event-triggered implementation, starting from initial condition \((12, 10, 5, 7)\) and setting \( x_1^* = 15 \) and \( x_2^* = 13 \). The corresponding results for the proposed decentralized event-triggered implementation are shown in Figure 2.4, and the results for a decentralized event-triggered implementation without adaptation, \( i.e., \) with \( \theta(k) = 0 \) for all \( k \in \mathbb{N} \), are shown in Figure 2.5.
For completeness, Figure 2.6 presents the evolution of adaptation vector $\theta$ for the adaptive decentralized event-triggered implementation. We can observe that, as expected, a centralized event-triggered implementation is far more efficient, in terms of time between updates, than a decentralized event-triggered implementation without adaptation. It is also clear that, although Algorithm 1 fails to recover the performance of the centralized event-triggered implementation exactly, it produces very good results. The results are even better if we look at the performance in terms of the number of executions which are presented in the legend of these plots. Finally we would like to remark that, although the times between updates in the three implementations can differ quite drastically, the three systems are...
2.6.2 Self triggered control

To illustrate the performance of the proposed self-triggered implementation for linear systems we borrow the Batch Reactor model from [WY01] with state space description:

stabilized producing almost undistinguishable state trajectories.
\[ \dot{\xi} = \begin{bmatrix} 1.38 & -0.20 & 6.71 & -5.67 \\ -0.58 & -4.29 & 0 & 0.67 \\ 1.06 & 4.27 & -6.65 & 5.89 \\ 0.04 & 4.27 & 1.34 & -2.10 \end{bmatrix} \xi + \begin{bmatrix} 0 & 0 \\ 5.67 & 0 \\ 1.13 & -3.14 \\ 1.13 & 0 \end{bmatrix} \chi \]

A state feedback controller placing the poles of the closed loop system at \( \{-3 + 1.2i, -3 - 1.2i, -3.6, -3.9\} \) is:

\[ K = \begin{bmatrix} 0.1006 & -0.2469 & -0.0952 & -0.2447 \\ 1.4099 & -0.1966 & 0.0139 & 0.0823 \end{bmatrix} \]

The closed loop has as decay rate \( \lambda_o = 0.41 \) and we set \( \lambda = 0.9\lambda_o \). The resulting minimum time for this selection of \( \lambda \) is \( t_{\text{min}} = 18ms \). The rest of design values were set to: \( t_{\text{max}} = 358ms \) and \( \Delta = 10ms \). With this design the complexity becomes \( M_s = 350 \) and \( M_t = 745 \).

Figure 2.7 presents the evolution of \( V(\xi(t)) \) (solid line) and the piecewise continuous function \( V(\xi(t_k))e^{-\lambda(t-t_k)} \) (dotted line) between seconds 1 and 2. The intersection of the dotted and solid lines (or the maximum value \( N_{\text{max}}\Delta \)) determines the inter-execution times \( \tau_k = \Gamma_d(\xi(t_k)) \). The actuation times \( t_k \) are marked with vertical dashed lines in Figure 2.7.

Figure 2.8 depicts the inter-execution times \( \tau_k \) generated by the self-triggered implementation in the absence of disturbances. The evolution of the Lyapunov function \( V(x) = (x^T P x)^{\frac{1}{2}} \) under disturbances (uniformly distributed bounded noise) with norms \( \|\delta\|_\infty = 1 \) and \( \|\delta\|_\infty = 10 \) are presented in figures 2.9, and 2.11 respectively. In both of those figures the ISS nature of the system can be appreciated. We also present the inter-execution times \( \tau_k \) generated by the self-triggered implementation under the presence of a disturbance with \( \|\delta\|_\infty = 1 \) in figure 2.10.
Figure 2.7: $V(\xi(t))$ (solid line) and $V(\xi(t_k))e^{-\lambda(t-t_k)}$ (dotted line) illustrating the triggering of new actuation.

Figure 2.8: Inter-execution times when no disturbance is present.

2.7 Discussion

We have presented two different techniques to implement controllers in an aperiodic fashion and argued their suitability for controller implementations over WSAN. These techniques, using feedback from the state of the plant, only demand actuation when required to stabilize the plant. This leads to savings on communication, and consequently on the energy consumed by the sensor nodes. In wireless nodes, besides the power consumed in computing or transmitting information, energy is consumed by keeping the radio communications module awake, i.e., there is a listening cost. In event-triggered implementations the sen-
sor nodes are always awake in order to react to updates triggered at other nodes. On the contrary, self-triggered implementations keep sensor nodes asleep in between controller updates, and thus reduce the incurred listening costs. This is possible in self-triggered implementations because, in them, the time of the next controller update is known immediately after the controller is updated. Hence, self-triggered techniques provide large savings in energy consumption by reducing listening time at the sensors. However, these savings are obtained at the expense of robustness to external disturbances. We have addressed this concern and shown that by imposing a maximum time between controller updates one
can provide guarantees of their response to disturbances. Imposing such upper limits on the time between updates may result in more frequent updates, which could offset the savings in listening costs. Improvements to these trade-offs can be achieved in a case by case analysis by looking at the specific constraints of the problem and possibly resorting to hybrid schemes, combining event-triggered and self-triggered techniques.

The proposed decentralized event-triggered implementation, while offering great savings, does not exploit all the possibilities an event-triggered framework offers to reduce communications. In particular, the techniques provided in Section 2.4 acquire measurements from all sensor nodes and update all actuation nodes synchronously. Ideally, one would like to request measurements and update actuators in an asynchronous fashion, accessing only the minimum number of sensors and actuators required to guarantee stability. Wang and Lemmon have taken steps in this direction and proposed a distributed event-triggered implementation which updates different actuator nodes independently of each other [WL09c]. However, their implementation is only applicable to control systems formed by weakly coupled subsystems. Our approach, while not updating inputs independently, does not rely on any internal coupling assumptions about
the system. Thus, our techniques can be used to complement the techniques in [WL09c] at the local subsystem level. The implementation of general dynamic controllers in event-triggered form, centralized or not, and the design of more efficient adaptation rules remain questions for future research. Finally, we would like to emphasize the low computational requirements of the proposed implementations, which makes them suitable for sensor/actuator networks with limited computation capabilities at the sensor level.

2.8 Appendix: Proofs

Proof of Theorem 2.5.3.

Space complexity: \((N_{\text{max}} - N_{\text{min}})\) vectors \(T(n)\) of size \(\frac{m(m+1)}{2}\) are needed to check the triggering condition at the different times \(t_k + n\Delta, n \in [N_{\text{min}}, N_{\text{max}}]\).

Time Complexity: The operation \(z(n) = T(n)\nu_2(\xi(t_k))\), requiring \(\frac{m(m+1)}{2}\) products and the same amount of additions, in the worst case needs to be performed \((N_{\text{max}} - N_{\text{min}})\) times. In addition, \(\frac{m(m+1)}{2}\) products are necessary to compute the embedding \(\nu_2(x)\). Moreover, \((N_{\text{max}} - N_{\text{min}})\) comparisons are required to enforce \(\tilde{h}(n, \xi(t_k)) \leq 0\). Adding all those terms proves the provided expression.

Proof of Lemma 2.5.4. It can be verified that \(h_c\) satisfies \(h_c(x, 0) = 0\) and \(\frac{\partial}{\partial t} \big|_{t=0} h_c(x, t) < 0, \forall x \in \mathbb{R}^m\), which, by continuity of \(h_c\), implies the existence of some \(\tau^*_{\text{min}}(x) > 0\) such that \(\Gamma_c(x) \geq \tau^*_{\text{min}}(x)\). Let us define the variables \(\eta(t) = \xi(t) - \xi(t_k), t \in [t_k, t_{k+1}[\) and \(\zeta = [\xi^T \eta^T]^T\). Note that at the controller update times \(\eta(t_k) = 0\). Under this new notation, system (2.12) with controller (2.13), in the absence of disturbances, can be rewritten as \(\dot{\zeta}(t) = F\zeta(t)\) with solution \(\zeta_y(t) = e^{Ft}y\), where \(y = [x^T 0^T]^T\). Let us denote by \(\hat{h}_c\) the map \(\hat{h}_c(y, t) = V(C\zeta_y(t)) - V(Cy)e^{-\lambda t}\).
While it is not possible to find $\Gamma_c$ in closed form, we can find its minimum value by means of the Implicit Function Theorem. Differentiating $\phi(x) = \hat{h}_c(y, \Gamma_c(x)) = 0$ with respect to the initial condition $x$ we obtain:

$$\frac{d\phi}{dx} = \frac{\partial \hat{h}_c}{\partial t} \bigg|_{t=\Gamma_c(x)} d\Gamma_c + \frac{\partial \hat{h}_c}{\partial y} dy dx = 0.$$ 

The extrema of the map $\Gamma_c$ are defined by the following equation:

$$\frac{d\Gamma_c}{dx} = - \left( \frac{\partial \hat{h}_c}{\partial t} \bigg|_{t=\Gamma_c(x)} \right)^{-1} \left( \frac{\partial \hat{h}_c}{\partial y} dy dx \right) = 0.$$ 

Hence, the extrema of $\Gamma_c$ satisfy either $\frac{\partial \hat{h}_c}{\partial y} dy dx (x,t) = 0$ for some $t \in \mathbb{R}^+$ or $\frac{d\Gamma_c}{dx} (x) = 0 \land \frac{\partial \hat{h}_c}{\partial x} \bigg|_{t=\Gamma_c(x)} = 0$. The latter case corresponds to situations in which for some $x$ the map $\hat{h}_c$ reaches zero exactly at an extremum, and thus can be disregarded as violations of the condition $h_c(t_k, t) \leq 0$. Combining $\frac{\partial \hat{h}_c}{\partial y} dy dx (\tau, x) = 0$ into matrix form we obtain:

$$M(\tau)x = 0. \quad (2.21)$$

The solution to this equation provides all extrema of the map $\Gamma_c(x)$ that incur a violation of $h_c(x,t) \leq 0$. Thus, the minimum $\tau$ satisfying (2.21) corresponds to the smallest time at which $h_c(x, \tau) = 0$, $\frac{\partial h_c}{\partial t} \bigg|_{t=\Gamma_c(x)} > 0$ can occur. Since the left hand side of (2.21) is linear in $x$, it is sufficient to check when the matrix has a nontrivial nullspace. Hence the equality (2.17). \qed

We introduce now a Lemma that will be used in the proof of Theorem 2.5.5.

**Lemma 2.8.1.** Consider system (2.12) and a positive definite function $V(x) = (x^T P x)^{\frac{1}{2}}$, $P > 0$. For any given $0 \leq T < \infty$ the following bound holds:

$$V(\xi_{x,0}(t)) \leq V(\xi_{x,0}(t)) + \gamma_{P,T} \|\delta\|_\infty, \forall t \in [0,T].$$
Proof. Applying the triangular inequality and using Lipschitz continuity of $V$ we have:

$$V(\xi_{x_0}(t)) = |V(\xi_{x_0}(t)) + V(\xi_{x_0}(t)) - V(\xi_{x_0}(t))|$$

$$\leq V(\xi_{x_0}(t)) + \frac{\lambda_M(P)}{\lambda_m^2(P)} |\xi_{x_0}(t) - \xi_{x_0}(t)|.$$ Integrating the dynamics of $\xi$ and after applying Hölder’s inequality one can conclude that:

$$|\xi_{x_0}(t) - \xi_{x_0}(t)| \leq \int_0^t |e^{Ap}| dr \|\delta\|_{\infty}.$$ And thus for all $t \in [0, T]$:

$$V(\xi_{x_0}(t)) \leq V(\xi_{x_0}(t)) + \frac{\lambda_M(P)}{\lambda_m^2(P)} \int_0^t |e^{Ap}| dr \|\delta\|_{\infty}.$$ 

Proof of Theorem 2.5.5. We start by proving that in the absence of disturbances the following bound holds:

$$|\xi_x(t_k + \tau)| \leq g(\Delta, N_{\text{max}}) |\xi_x(t_k)| e^{-\lambda \tau}, \forall \tau \geq 0. \quad (2.22)$$

Let $W(x) = x^T Px$ and use $W(t)$ to denote $W(\xi_{x_k}(t))$, with $\xi$ determined by (2.12), (2.13), and $\tau_k = \Gamma_d(\xi(t_k))$. By explicitly computing $\dot{W}(t)$ one obtains:

$$\dot{W}(t) = \left[(P^T \xi(t))^T (P^T \xi(t))\right] G \left[(P^T \xi(t))^T (P^T \xi(t))\right]^T,$$

for $t \in [t_k, t_{k+1}]$, and thus the following bounds hold:

$$\mu(W(t) + W(t_k)) \leq \dot{W}(t) \leq \rho(W(t) + W(t_k)).$$

for $t \in [t_k, t_{k+1}]$. After integration, one can bound the trajectories of $W(t)$, when $t + s$ belongs to the interval $[t_k, t_{k+1}]$, as:

$$W(t + s) \leq e^{\beta s} W(t) + W(t_k) (e^{\beta s} - 1),$$

$$W(t + s) \geq e^{\mu s} W(t) + W(t_k) (e^{\mu s} - 1).$$
Let us denote $t_k + n\Delta$ by $r_n$ for succinctness of the expressions that follow. An upper bound for $W(t)$ valid for $t \in [r_n, r_{n+1}]$ is then provided by:

$$W(r_n + s) \leq \begin{cases} 
  e^{\rho s}W(r_n) + W(t_k)(e^{\rho s} - 1), & s \in [0, s^*] \\
  e^{\mu(s-\Delta)}W(r_n + \Delta) + W(t_k)(e^{\mu(s-\Delta)} - 1), & s \in [s^*, \Delta]. 
\end{cases}$$

The maximum for the bound of $W(r_n + s)$ for $s \in [0, \Delta]$, is attained at the point at which the two branches of the bound meet, i.e., at $s = s^*$, as the first branch is monotonically increasing in $s$, and the second branch monotonically decreasing. The point $s^*$ can be computed as:

$$s^* = \frac{1}{\rho - \mu} \log \left( \frac{W(r_{n+1}) + W(t_k)}{W(r_n) + W(t_k)} \right) + \frac{\mu \Delta}{\mu - \rho}$$

and thus $W(r_n + s^*)$ can be bounded as:

$$W(r_n + s^*) \leq -W(t_k) + e^{\frac{\mu \Delta}{\mu - \rho}} \left( (W(r_n) + W(t_k))^{\frac{\mu}{\mu - \rho}} (W(r_{n+1}) + W(t_k))^{\frac{\rho}{\rho - \mu}} \right)$$

which is monotonically increasing on $W(r_n)$, $W(r_{n+1})$, and $W(t_k)$. If $S(t) = W(t_k)e^{-2\lambda(t-t_k)}$, it follows:

$$W(r_n + s^*) \leq -S(t_k) + e^{\frac{\rho s^*}{\rho - \mu}} \left( (S(r_n) + S(t_k))^{\frac{\mu}{\mu - \rho}} (S(r_{n+1}) + S(t_k))^{\frac{\rho}{\rho - \mu}} \right)$$

where we used the fact that, if $\tau_{min} \leq \tau^*_d$, $\Gamma_d$ enforces (in the absence of disturbances) $W(r_n) \leq S(r_n)$ for all $n \in \mathbb{N}$, $n \leq n_k$. From the previous expression we can obtain $W(r_n + s^*) \leq \tilde{g}(\Delta, n)S(r_n + s^*)$ where:

$$\tilde{g}(\Delta, n) = -e^{2\lambda(n\Delta + s^*)} + e^{\frac{\rho s^*}{\rho - \mu}} (e^{2\lambda s^*} (1 + e^{2\lambda n\Delta}))^{\frac{\mu}{\mu - \rho}} (e^{-2\lambda(\Delta-s^*)} + e^{2\lambda(n\Delta+s^*)})^{\frac{\rho}{\rho - \mu}}.$$ 

The value of $s^*$ can be further bounded to obtain a simpler expression:

$$s^* \leq \frac{\mu \Delta}{\mu - \rho}.$$ 

Using this bound for $s^*$ and letting $n$ take its maximum possible value $n = N_{\max} - 1$, the following chain of inequalities holds:

$$\rho P \tilde{g}(\Delta, n)^{\frac{1}{2}} \leq \rho P \tilde{g}(\Delta, N_{\max} - 1)^{\frac{1}{2}} \leq g(\Delta, N_{\max})$$
for all $n \in [0, N_{\text{max}}]$, which leads to the bound:

$$W \frac{1}{2}(t) \leq \rho_P^{-1} g(\Delta, N_{\text{max}}) S \frac{1}{2}(t). \quad (2.23)$$

Note that (2.23) does not depend on $t_k$ or $n$. Finally, apply the bounds:

$$\lambda_M^2(P)|x| \leq V(x) = \sqrt{x^TPx} \leq \lambda_M^2(P)|x|. \quad (2.24)$$

to obtain (2.22). From Lemma 2.8.1, and the condition enforced by the self-triggered implementation we have:

$$V(\xi(t_{k+1})) \leq V(\xi(t_k))e^{-\lambda t} + \gamma_P(\|\delta\|_{\infty}).$$

Iterating the previous equation it follows:

$$V(\xi(t_k)) \leq e^{-\lambda(t_k-t_o)}V(\xi(t_0)) + \gamma_P(\|\delta\|_{\infty}) \sum_{i=0}^{k-1} e^{-\lambda \tau_i}.$$

Assuming, without loss of generality, that $t_o = 0$, the following bound also holds:

$$|\xi_x(t_k)| \leq \rho_P|x|e^{-\lambda t_k} + \frac{\lambda_m^{-\frac{1}{2}}(P) \gamma_P(\|\delta\|_{\infty})}{1 - e^{-\lambda t}}. \quad (2.25)$$

where we used (2.24). From (2.22) and Lemma 2.8.1 one obtains:

$$|\xi_x(t_k + \tau)| \leq g(\Delta, N_{\text{max}})|\xi_x(t_k)|e^{-\lambda \tau} + \gamma_I(\|\delta\|_{\infty}), \quad (2.26)$$

for all $\tau \in [0, N_{\text{max}} \Delta]$. Combining (2.25) and (2.26) results in:

$$|\xi_x(t_k + \tau)| \leq g(\Delta, N_{\text{max}})\rho_P|x|e^{-\lambda(t_k + \tau)} + e^{-\lambda \tau} \gamma_P(\|\delta\|_{\infty}) \frac{\lambda_m^{-\frac{1}{2}}(P) g(\Delta, N_{\text{max}})}{1 - e^{-\lambda t}} + \gamma_I(\|\delta\|_{\infty}),$$

and after denoting $t_k + \tau$ by $t$ we can further bound:

$$|\xi_x(t)| \leq g(\Delta, N_{\text{max}})\rho_P|x|e^{-\lambda t} + \gamma_P(\|\delta\|_{\infty}) \frac{\lambda_m^{-\frac{1}{2}}(P) g(\Delta, N_{\text{max}})}{1 - e^{-\lambda t}} + \gamma_I(\|\delta\|_{\infty}),$$

which is independent of $k$ and concludes the proof. \qed
CHAPTER 3

Correct-by-design synthesis of embedded controllers

3.1 Introduction

Embedded controllers are digital implementations of control systems in electronic devices working with little or no human supervision. Many of these embedded controllers are responsible for the adequate operation of life critical systems. As such, provably correct operation, and safety in particular, is a typical design requirement. Much work has been devoted by the computer science community to the verification of software and hardware correctness. Techniques and tools capable of providing proofs, or counter-examples, of the satisfaction of given specifications are already available. However, these methods cannot be directly applied to the verification of systems described by continuous dynamical models, such as differential equations. Software and hardware are typically modeled as finite state machines. These models contain a finite number of states and their dynamics are governed by discrete transitions between those states. Thus, most of the tools developed for correctness verification require a description of the systems to be verified using such models.

Control systems are most often described by differential equations. Symbolic abstractions are simpler descriptions of such control systems, typically with
finitely many states, in which each symbolic state represents a collection or aggregate of states in the control system. Once such abstractions are available, the methodologies and tools developed in computer science for verification purposes can be employed to control systems, via these abstractions. Most embedded control systems are of hybrid nature: on the one hand the physical plant they control is generally a continuous time dynamical system; on the other hand, these controllers are almost always implemented on digital hardware and interact with other software, both exhibiting discrete time dynamics. The complex interactions between continuous and discrete dynamics on embedded controller implementations make the analysis and design of such systems arduous. One can resort to symbolic abstractions of the control system itself, i.e., the continuous dynamics, and compose the resulting model with another appropriate symbolic model describing the hardware/software platform. In this way it is possible to obtain a symbolic model incorporating both dynamics from the physical plant and the implementation platform.

While most of the tools available are aimed at the verification of already designed systems, a new paradigm has started to gain momentum among researchers: correct-by-design synthesis. Verification tools can prove or disprove the correctness of a given design, but, if the design is incorrect they only provide with counter examples exploiting flaws in the design. Yet, one is left with a redesign problem that often can be as complicated as the original design problem. Correct-by-design synthesis on the other hand aims at the automatic generation of systems satisfying a provided specification. Thus, correct-by-design synthesis approaches are more general than verification. In particular, any verification problem can be casted as a design problem. The verification of a system can be performed by checking if a correct-by-design synthesis returns a trivial controller, which would imply the correctness of the original design.
Most verification and correct-by-design synthesis techniques address qualitative specifications: behaviors of the system either are included or they are not included in the given specification. However, in may cases one is also interested in quantitative descriptions associating costs or utility values to different behaviors. Associating such quantitative properties to a system’s behavior enables the specification and solution of optimal control problems. In this chapter I propose methods to construct symbolic abstractions of control systems, and we analyze the correct-by-design synthesis problem when qualitative specifications are included, namely time-optimal control problems. Moreover, we introduce a toolbox for Matlab capable of constructing abstractions for continuous time control systems and synthesizing controllers for several specifications, including time-optimal reachability.

The analysis and design methodologies described in this chapter have been mainly collected from the book [Tab09], and my contributions from the publications [ZPM10], [MT10c] and [MDT10].

3.1.1 Previous work

The analysis and design of controllers for hybrid systems, the mathematical models employed to describe embedded systems, has spurred a great amount of research. A large part of this research has been devoted to the study of symbolic abstractions for control systems. By resorting to such abstractions, computational tools developed for discrete-event systems [KG95, CL99] and games on automata [AHM01, MNA03, AVW03] can be employed to synthesize controllers satisfying specifications difficult to enforce with conventional control design methods. Examples of such specifications include requirements given by means of temporal-logics, ω-regular languages, or automata on infinite strings. In prac-
tice, most solutions to such problems are obtained through hierarchical designs with supervisory controllers on the top layers. Such designs are usually the result of an ad-hoc process for which correctness guarantees are hard to obtain. Moreover, these kinds of designs require a certain level of insight that just the most experienced system designers posses. The use of symbolic control [GP09, PGT08, EFJ06] has emerged as an alternative to ad-hoc designs. Early efforts to construct symbolic abstractions for continuous dynamics can be found on the study of timed automata [AD90], rectangular hybrid automata [HKP98], and o-minimal hybrid systems [LPS00, BM05]. Other results rely on dynamical consistency properties [CW98], natural invariants of the control system [KAS00], l-complete approximations [MRO02], and quantized inputs and states [FJL02, BMP02]. More recent results include the work on piecewise-affine and multi-affine systems [HCS06, BH06] and the study of convexity of reachable sets to improve accuracy of symbolic abstractions based on reachability analysis [Rei09]. Many tools have also been developed for the analysis of hybrid systems. Most of these tools, such as Ariadne [Ari], PHAVer [PHA], KeYmaera [KeY], Checkmate [Che], and HybridSAL [Hyba], focus on verification problems. Tools for the synthesis of controllers are more recent and include LTLCon [LTL] for linear control systems and the Hybrid Toolbox [Hybb] for piece-wise affine hybrid systems.

A widely used method to analyze abstractions relies on the study of systems behaviors. The study of inclusions or equality of the observed behaviors produced by different models provides a mean to analyze the value of a given abstraction. A popular notion to describe such behavioral relations is that of simulation and bisimulation relations [Mil89]. However, requiring exact equalities between the observed behaviors of a control system and the behaviors of an abstraction is often too strong, and results in a constricted applicability of these methods. This
problem was solved in [GP07] through the introduction of approximate simulation
and bisimulation relations to relate the symbolic abstractions to the original
control systems. These approximate notions only require the observed behaviors
from the control system and its abstraction to be close enough to each other,
but not necessarily equal. Making use of these relaxed relations, constructions
of abstractions were successfully applied to incrementally input-to-state stable
systems with and without disturbances in [PGT08, PT09] and to incrementally
stable switched systems in [GPT09].

The work on symbolic abstractions has been mostly applied to the solution
of problems with qualitative specifications, \textit{i.e.}, problems in which behaviors are
treated as desired or forbidden. Their relative success to solve such problems
advocates the use of symbolic models also for problems with quantitative speci-
fications, \textit{i.e.}, problems in which behaviors have costs or values associated that
need to be optimized. Since the illustrious seminal contributions in the 50’s by
Pontryagin [Pon59] and Bellman [Bel52], the design of optimal controllers has
remained a standing quest of the controls community. Despite the several ad-
vances since then, solving optimal control problems with complex geometries on
the state space, constraints in the input space, and/or complex dynamics is still
a daunting task. Several symbolic techniques to solve such complex optimization
problems have been developed. A common method in the literature has been
to discretize the dynamics and apply optimal search algorithms on graphs such
as Dijkstra’s algorithm [GJ09, BDD05, TI08]. Other techniques can be found in-
cluding Mixed (Linear or Quadratic) Integer Programing [KSF08] and the use of
SAT-solvers [BG06].
3.1.2 Contributions

In this chapter we present symbolic abstractions for a general class of control systems and we study their suitability to synthesize controllers enforcing both qualitative and quantitative specifications. Qualitative specifications require the controller to preclude certain undesired trajectories from the system to be controlled. The term qualitative refers to the fact that all the desired trajectories are treated as being equally good. In many practical applications, while there are plant trajectories that must be eliminated, there is also a need to select the best of the remaining trajectories. Typically, the best trajectory is specified by means of a cost or utility associated to each trajectory, a qualitative property. The control design problem then requires the removal of the undesirable trajectories and the selection of the minimum cost or maximum utility trajectory. As a first step towards the objective of synthesizing controllers enforcing qualitative and quantitative objectives, we consider in the present chapter the synthesis of time-optimal controllers for reachability specifications.

We start by showing in Section 3.3 that symbolic models of control systems without relying on stability assumptions, as was the case in previous work [GP09, PGT08], exist. The stability assumptions on the control systems are substituted by a milder requirement termed *incremental forward completeness*. This is an incremental version of forward completeness and is satisfied by any smooth control system on a compact set. The symbolic models constructed under this assumption are shown to be alternatingly approximately simulated by the control system and approximately simulate the control system. A thorough discussion of these results can be found in [ZPM10].

Section 3.4 introduces the main theoretical contribution of the chapter: we show that time-optimality information can be transferred from a system $S_a$ to
a system $S_b$, when system $S_a$ is related to system $S_b$ by an approximate (alternating) simulation relation. Hence, the analysis of optimality considerations is decoupled from the design of algorithms extracting a discretization $S_a$ from the original system $S_b$. Using this result, we show how to construct an approximately time-optimal controller for system $S_b$ from a time-optimal controller for system $S_a$. Rather than showing that by using finer discretizations one obtains controllers that are arbitrarily close to the optimal one [BDD05], the technique we present efficiently computes an approximate solution and establishes how much it deviates from the true optimal cost or utility.

The proposed results are independent of the specific techniques employed in the construction of symbolic abstractions provided that the existence of approximately (alternating) simulations relations is established. The specific constructions reported in Section 3.3 show that our assumptions can be met for a large class of systems, thus making the use of the methods we propose widely applicable. Furthermore, efficient algorithms and data structures from computer science can be used to implement the proposed techniques, see for example the recent work on optimal synthesis [BCH09]. In particular, the toolbox Pessoa, introduced in Section 3.5, uses Binary Decision Diagrams (BDD’s) [Weg00] to store systems modeling both plants and controllers. The fact that BDD’s can be used to automatically generate hardware [BGJ07] or software [BCG99] implementations of the controllers makes them specially attractive. The Matlab toolbox Pessoa [MDT10] represents the most practical contribution of this chapter and, arguably, this thesis. What sets Pessoa apart from the existing tools for the analysis of hybrid systems is the nature of the abstractions (approximate simulations and bisimulations) and the broad class of systems admitting such abstractions (linear, nonlinear, and switched [Tab09]).
3.2 Preliminaries

3.2.1 Notation

Let us start by introducing some notation that will be used throughout the present chapter. We denote by \( \mathbb{N} \) the natural numbers including zero and by \( \mathbb{N}^+ \) the strictly positive natural numbers. With \( \mathbb{R}^+ \) we denote the strictly positive real numbers, and with \( \mathbb{R}_0^+ \) the positive real numbers including zero. The identity map on a set \( A \) is denoted by \( 1_A \). If \( A \) is a subset of \( B \) we denote by \( ı_A : A \hookrightarrow B \) or simply by \( ı \) the natural inclusion map taking any \( a \in A \) to \( ı(a) = a \in B \). The closed ball centered at \( x \in \mathbb{R}^n \) with radius \( \varepsilon \) is defined by \( B_\varepsilon(x) = \{ y \in \mathbb{R}^n \mid \| x - y \| \leq \varepsilon \} \). We denote by \( \text{int}(A) \) the interior of a set \( A \).

The symbols \( \mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{R}^+ \) and \( \mathbb{R}_0^+ \) denote the set of natural, integer, real, positive, and nonnegative real numbers, respectively. The symbol \( I_m \) denotes the identity matrix on \( \mathbb{R}^m \). Given a vector \( x \in \mathbb{R}^n \), we denote by \( x_i \) the \( i \)-th element of \( x \), by \( \| x \| \) the infinity norm of \( x \), and by \( \| x \|_2 \) the Euclidean norm of \( x \); we recall that \( \| x \| = \max\{|x_1|, |x_2|, ..., |x_n|\} \), and \( \| x \|_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \), where \( |x_i| \) denotes the absolute value of \( x_i \). A normed vector space \( V \) is a vector space equipped with a norm \( \| \cdot \| \), as is well-known this induces the metric \( d(x, y) = \| x - y \|, \ x, y \in V \). Given an essentially bounded function \( \delta : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \) we denote by \( \| \delta \|_\infty \) its \( L_\infty \) norm, i.e., \( \| \delta \|_\infty = (\text{ess sup}_{t \in \mathbb{R}_0^+} \{ |\delta(t)| \} < \infty \). For any \( A \subseteq \mathbb{R}^n \) and \( \mu \in \mathbb{R} \) we define the set \( [A]_\mu = \{ a \in A \mid a_i = k_i \mu, k_i \in \mathbb{Z}, i = 1, ..., n \} \). The set \( [A]_\mu \) will be used as an approximation of the set \( A \) with precision \( \mu \).

Geometrically, for any \( \mu \in \mathbb{R}^+ \) and \( \lambda \geq \mu/2 \) the collection of sets \( \{ B_\lambda(q) \}_{q \in [\mathbb{R}^n]_\mu} \) is a covering of \( \mathbb{R}^n \). Abusing notation, we also denote for a set \( A \) its cardinality by \( |A| \).

A continuous function \( \gamma : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \), is said to belong to class \( \mathcal{K} \) if it is strictly increasing and \( \gamma(0) = 0 \); \( \gamma \) is said to belong to class \( \mathcal{K}_\infty \) if \( \gamma \in \mathcal{K} \) and \( \gamma(r) \to \infty \) as \( r \to \infty \).
as \( r \to \infty \). We identify a relation \( R \subseteq A \times B \) with the map \( R : A \to 2^B \) defined by \( b \in R(a) \) iff \( (a,b) \in R \). For a set \( S \in A \) the set \( R(S) \) is defined as \( R(S) = \{ b \in B : \exists a \in S, (a,b) \in R \} \). Also, \( R^{-1} \) denotes the inverse relation defined by \( R^{-1} = \{ (b,a) \in B \times A : (a,b) \in R \} \). We also denote by \( d : X \times X \to \mathbb{R}_0^+ \) a metric in the space \( X \) and by \( \pi_X : X_a \times X_b \times U_a \times U_b \to X_a \times X_b \) the projection sending \( (x_a, x_b, u_a, u_b) \in X_a \times X_b \times U_a \times U_b \) to \( (x_a, x_b) \in X_a \times X_b \).

### 3.2.2 Systems and control systems

In this chapter we use the mathematical notion of *systems* to model dynamical phenomena. This notion is formalized in the following definition:

**Definition 3.2.1 (System [Tab09]).** A system \( S \) is a sextuple \( (X, X_0, U, \to, Y, H) \) consisting of:

- a set of states \( X \);
- a set of initial states \( X_0 \subseteq X \);
- a set of inputs \( U \);
- a transition relation \( \to \subseteq X \times U \times X \);
- a set of outputs \( Y \);
- an output map \( H : X \to Y \).

A system is said to be:

- metric, if the output set \( Y \) is equipped with a metric \( d : Y \times Y \to \mathbb{R}_0^+ \);
- countable, if \( X \) is a countable set;
- finite, if \( X \) is a finite set.

We use the notation \( x \xrightarrow{u} y \) to denote \( (x, u, y) \in \to \). For a transition \( x \xrightarrow{u} y \), state \( y \) is called a \( u \)-successor, or simply successor. We denote the set
of \( u \)-successors of a state \( x \) by \( \text{Post}_u(x) \). If for all states \( x \) and inputs \( u \) the sets \( \text{Post}_u(x) \) are singletons (or empty sets) we say the system \( S \) is deterministic. If, on the other hand, for some state \( x \) and input \( u \) the set \( \text{Post}_u(x) \) has cardinality greater than one, we say that system \( S \) is non-deterministic. Furthermore, if there exists some pair \( (x, u) \) such that \( \text{Post}_u(x) = \emptyset \) we say the system is blocking, and otherwise non-blocking. We also use the notation \( U(x) \) to denote the set \( U(x) = \{ u \in U | \text{Post}_u(x) \neq \emptyset \} \).

Nondeterminism arises for a variety of reasons such as modeling simplicity. Nevertheless, to every nondeterministic system \( S_a \) we can associate a deterministic system \( S_{d(a)} \) by extending the set of inputs:

**Definition 3.2.2** (Associated deterministic system). The deterministic system \( S_{d(a)} = (X_a, X_{a0}, U_{d(a)}, y_0, \text{H}_a) \) associated with a given system \( S_a = (X_a, X_{a0}, U_a, y_0, \text{H}_a) \), is defined by:

- \( U_{d(a)} = U_a \times X_a \);
- \( x \xrightarrow{(u, x')} y' \) if there exists \( x \xrightarrow{u} y \) in \( S_a \).

Sometimes we need to refer to the possible sequences of outputs that a system can exhibit. We call these sequences of outputs behaviors. Formally, behaviors are defined as follows:

**Definition 3.2.3** (Behaviors [Tab09]). For a system \( S \) and given any state \( x \in X \), a finite behavior generated from \( x \) is a finite sequence of transitions:

\[
y_0 \longrightarrow y_1 \longrightarrow y_2 \longrightarrow \cdots \longrightarrow y_{n-1} \longrightarrow y_n
\]

such that \( y_0 = H(x) \) and there exists a sequence of states \( \{x_i\} \), and a sequence of inputs \( \{u_i\} \) satisfying: \( H(x_i) = y_i \) and \( x_{i-1} \xrightarrow{u_{i-1}} x_i \) for all \( 0 \leq i < n \).
An infinite behavior generated from $x$ is an infinite sequence of transitions:

$$y_0 \rightarrow y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow \ldots$$

such that $y_0 = H(x)$ and there exists a sequence of states $\{x_i\}$, and a sequence of inputs $\{u_i\}$ satisfying: $H(x_i) = y_i$ and $x_{i-1} \xrightarrow{u_{i-1}} x_i$ for all $i \in \mathbb{N}$.

By $\mathcal{B}_x(S)$ and $\mathcal{B}^\omega_x(S)$ we denote the set of finite and infinite external behaviors generated from $x$, respectively. Sometimes we use the notation $\mathbf{y} = y_0y_1y_2 \ldots y_n$, to denote external behaviors, and $\mathbf{y}(k)$ to denote the $k$-th output of the behavior, i.e., $y_k$. A behavior $\mathbf{y}$ is said to be maximal if there is no other behavior containing $\mathbf{y}$ as a prefix.

The class of control systems considered in this chapter is formalized in the following definition:

**Definition 3.2.4** (Control system). A control system is a quadruple:

$$\Sigma = (\mathbb{R}^n, U, \mathcal{U}, f),$$

where:

- $\mathbb{R}^n$ is the state space;
- $U \subseteq \mathbb{R}^m$ is the input space;
- $\mathcal{U}$ is a subset of the set of all functions of time from intervals of the form $]a, b[$ to $\mathcal{U}$ with $a < 0$, $b > 0$, and satisfying the following Lipschitz assumption: there exists a constant $K \in \mathbb{R}^+$ such that $\|v(t) - v(t')\| \leq K|t - t'|$ for all $v \in \mathcal{U}$ and for all $t, t' \in ]a, b[$;
- $f : \mathbb{R}^n \times U \to \mathbb{R}^n$ is a continuous map satisfying the following Lipschitz assumption: for every compact set $Q \subset \mathbb{R}^n$, there exists a constant $Z \in \mathbb{R}^+$ such that $\|f(x, u) - f(y, u)\| \leq Z\|x - y\|$ for all $x, y \in Q$ and all $u \in \mathcal{U}$. 

55
A curve $\xi : ]a, b[ \to \mathbb{R}^n$ is said to be a \textit{trajectory} of $\Sigma$ if there exists $v \in U$ satisfying:

$$\dot{\xi}(t) = f(\xi(t), v(t)), \quad (3.1)$$

for almost all $t \in ]a, b[$. Although we have defined trajectories over open domains, we shall refer to trajectories $\xi : [0, \tau] \to \mathbb{R}^n$ defined on closed domains $[0, \tau], \tau \in \mathbb{R}^+$ with the understanding of the existence of a trajectory $\xi' : ]a, b[ \to \mathbb{R}^n$ such that $\xi = \xi'|[0,\tau]$. We also write $\xi_{x,v}(\tau)$ to denote the point reached at time $\tau$ under the input $v$ from initial condition $x$; this point is uniquely determined, since the assumptions on $f$ ensure existence and uniqueness of trajectories [Son98]. We also denote an autonomous system $\Sigma$ with no control inputs by $\Sigma = (\mathbb{R}^n, f)$. A control system $\Sigma$ is said to be forward complete if every trajectory is defined on an interval of the form $]a, \infty[$. Sufficient and necessary conditions for a system to be forward complete can be found in [AS99]. A control system $\Sigma$ is said to be smooth if $f$ is an infinitely differentiable function of its arguments.

Given a control system $\Sigma = (\mathbb{R}^n, U, U, f)$ and time discretization parameter $\tau \in \mathbb{R}^+$, we associate the following system to $\Sigma$:

$$S_\tau(\Sigma) := (X_\tau, X_\tau, U_\tau, X_\tau, Y_\tau, H_\tau),$$

where:

- $X_\tau = \mathbb{R}^n$;
- $U_\tau = \{v_\tau \in U \mid \text{the domain of } v_\tau \text{ is } [0, \tau]\}$;
- $x_\tau \xrightarrow{v_\tau}{x_\tau'}$ if there exists a trajectory $\xi : [0, \tau] \to \mathbb{R}^n$ of $\Sigma$ satisfying $\xi_{x_\tau,v_\tau}(\tau) = x_\tau'$;
- $Y_\tau = \mathbb{R}^n$;
- $H_\tau = 1_{\mathbb{R}^n}$. 

56
The above system can be thought of as the time discretization of the control system $\Sigma$. In Section 3.3, we show how to obtain a countable abstraction model for $S_\tau(\Sigma)$.

Although we have defined trajectories over open domains, we shall refer to trajectories $\xi : [0, \tau] \rightarrow \mathbb{R}^n$ defined on closed domains $[0, \tau]$, $\tau \in \mathbb{R}^+$ with the understanding of the existence of a trajectory $\xi' : [a, b] \rightarrow \mathbb{R}^n$ such that $\xi = \xi'|_{[0, \tau]}$. We also write $\xi_{x\upsilon}(t)$ to denote the point reached at time $t \in [0, \tau]$ under the input $\upsilon$ from initial condition $x$; this point is uniquely determined, since the assumptions on $f$ ensure existence and uniqueness of trajectories.

### 3.2.3 Incremental forward completeness

The results presented in this chapter require certain assumptions that we introduce in this section. We start by recalling the notion of incremental global asymptotic stability.

**Definition 3.2.5** (Incremental global asymptotic stability [Ang02]). *A control system $\Sigma$ is incrementally globally asymptotically stable ($\delta$-GAS) if it is forward complete and there exists a KL function $\beta$ such that for any $t \in \mathbb{R}^+_0$, any $x, x' \in \mathbb{R}^n$ and any $\upsilon \in \mathcal{U}$ the following condition is satisfied:*

$$\|\xi_{x\upsilon}(t) - \xi_{x'\upsilon}(t)\| \leq \beta(\|x - x'\|, t).$$

(3.2)

Whenever the origin is an equilibrium point for $\Sigma$, $\delta$-GAS implies global asymptotic stability (GAS).

**Definition 3.2.6** (Incremental input-to-state stability [Ang02]). *A control system $\Sigma$ is incrementally input-to-state stable ($\delta$-ISS) if it is forward complete and there exist a KL function $\beta$ and a $K_\infty$ function $\gamma$ such that for any $t \in \mathbb{R}^+_0$, any
x, x′ ∈ ℜ^n, and any v, v′ ∈ U the following condition is satisfied:

\[ \|\xi_{xv}(t) - \xi_{x'v'}(t)\| \leq \beta(\|x - x'\|, t) + \gamma(\|v - v'\|_\infty). \] (3.3)

By observing (3.2) and (3.3), it is readily seen that δ-ISS implies δ-GAS while the converse is not true in general. Moreover, if the origin is an equilibrium point for Σ, δ-ISS implies input-to-state stability (ISS). We now describe a weaker concept that is satisfied even in the absence of stability.

**Definition 3.2.7** (Incremental forward completeness). A control system Σ is incrementally forward complete (δ-FC) if there exist continuous functions \( \beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+ \) and \( \gamma : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+ \) such that for every \( s \in \mathbb{R}_0^+ \), the functions \( \beta(\cdot, s) \) and \( \gamma(\cdot, s) \) belong to class \( \mathcal{K}_\infty \), and for any \( t \in \mathbb{R}_0^+ \), any \( x, x' \in \mathbb{R}^n \) and any \( v, v' \in U \) the following condition is satisfied:

\[ \|\xi_{xv}(t) - \xi_{x'v'}(t)\| \leq \beta(\|x - x'\|, t) + \gamma(\|v - v'\|_\infty, t). \] (3.4)

Incremental forward completeness requires the distance between two arbitrary trajectories to be bounded by the sum of two terms capturing the mismatch between the initial conditions and the mismatch between the inputs as shown in (3.4).

As an example, for a linear control system:

\[ \dot{\xi} = A\xi + Bv, \quad \xi(t) \in \mathbb{R}^n, \quad v(t) \in U \subseteq \mathbb{R}^m, \]

the functions \( \beta \) and \( \gamma \) can be chosen as:

\[ \beta(r, t) = \|e^{At}\| r; \quad \gamma(r, t) = \left( \int_0^t \|e^{As}B\| ds \right) r, \] (3.5)

where \( \|e^{At}\| \) denotes the infinity norm\(^1\) of \( e^{At} \). From (3.3) and (3.4), we can immediately see that δ-ISS implies δ-FC. However, the converse is not true, in

---

\(^1\)For \( M = \{m_{ij}\} \in \mathbb{R}^{n \times m} \), the infinity norm of \( M \) is, \( \|M\| := \max_{1 \leq i \leq n} \sum_{j=1}^{m} |m_{ij}| \).
general, since the function \( \beta \) in (3.4) is not required to be a decreasing function of \( t \) and the function \( \gamma \) in (3.4) is allowed to depend on \( t \) while this is not the case in (3.3). Whenever the origin is an equilibrium point for \( \Sigma \), the choice \( x' = 0, v' = 0 (\xi_{x'v'} = 0) \) results in the estimate \( \| \xi_{xv}(t) \| \leq \beta(\|x\|, t) + \gamma(\|v\|_\infty, t) \) which is shown in [AS99] to be equivalent to forward completeness of \( \Sigma \).

### 3.2.4 Systems relations

The results we prove build upon certain simulation relations that can be established between systems. The first relation explains how a system can simulate another system.

**Definition 3.2.8** (Approximate simulation relation [Tab09]). Let \( S_a \) and \( S_b \) be metric systems with \( Y_a = Y_b \) and let \( \varepsilon \in \mathbb{R}^+_0 \). A relation \( R \subset X_a \times X_b \) is an \( \varepsilon \)-approximate simulation relation from \( S_a \) to \( S_b \) if the following three conditions are satisfied:

(i) for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);

(ii) for every \( (x_a, x_b) \in R \) we have \( d(H_a(x_a), H_b(x_b)) \leq \varepsilon \);

(iii) for every \( (x_a, x_b) \in R \) we have that \( x_a \xrightarrow{u_a} x'_a \) in \( S_a \) implies the existence of \( x_b \xrightarrow{u_b} x'_b \) in \( S_b \) satisfying \( (x'_a, x'_b) \in R \).

We say that \( S_a \) is \( \varepsilon \)-approximately simulated by \( S_b \) or that \( S_b \) \( \varepsilon \)-approximately simulates \( S_a \), denoted by \( S_a \preceq^\varepsilon S_b \), if there exists an \( \varepsilon \)-approximate simulation relation from \( S_a \) to \( S_b \).

When \( S_a \preceq^\varepsilon S_b \), system \( S_b \) can replicate the behavior of system \( S_a \) by starting at a state \( x_{b0} \in X_{b0} \) related to any initial state \( x_{a0} \in X_{a0} \) and by replicating every transition in \( S_a \) with a transition in \( S_b \) according to (3). It then follows from (2)
that the resulting behaviors will be the same up to an error of $\varepsilon$. If $\varepsilon = 0$ the second condition implies that two states $x_a$ and $x_b$ are related whenever their outputs are equal, i.e., $(x_a, x_b) \in R$ implies $H(x_a) = H(x_b)$, and we say that the relation is an exact simulation relation. When nondeterminism is regarded as adversarial, the notion of approximate simulation can be modified by explicitly accounting for nondeterminism.

Symmetrizing the notion of simulation we obtain the notion of bisimulation, which we report hereafter.

**Definition 3.2.9** (Approximate bisimulation relation). Let $S_a$ and $S_b$ be metric systems with the same output sets $Y_a = Y_b$ and metric $d$, and consider a precision $\varepsilon \in \mathbb{R}^+$. A relation $R \subseteq X_a \times X_b$ is said to be an $\varepsilon$-approximate bisimulation relation between $S_a$ and $S_b$, if the following two conditions are satisfied:

(i) $R$ is an $\varepsilon$-approximate simulation relation from $S_a$ to $S_b$;
(ii) $R^{-1}$ is an $\varepsilon$-approximate simulation relation from $S_b$ to $S_a$.

System $S_a$ is $\varepsilon$-approximate bisimilar to $S_b$, denoted by $S_a \cong^\varepsilon S_b$, if there exists an $\varepsilon$-approximate bisimulation relation $R$ between $S_a$ and $S_b$.

For nondeterministic systems we need to consider relationships that explicitly capture the adversarial nature of nondeterminism. It was illustrated in [PT09] that the preceding notions of simulation and bisimulation are not appropriate for symbolic control design on nondeterministic systems. In the following, we report the notions of alternating approximate simulation and bisimulation which, as illustrated in [PT09], are appropriate for nondeterministic systems.

**Definition 3.2.10** (Approximate alternating simulation relation [Tab09]). Let $S_a$ and $S_b$ be metric systems with $Y_a = Y_b$ and let $\varepsilon \in \mathbb{R}^+$. A relation $R \subseteq X_a \times X_b$
is an \( \varepsilon \)-approximate alternating simulation relation from \( S_a \) to \( S_b \) if the following three conditions are satisfied:

(i) for every \( x_{a0} \in X_a \) there exists \( x_{b0} \in X_b \) with \( (x_{a0}, x_{b0}) \in R \);

(ii) for every \( (x_a, x_b) \in R \) we have \( \text{d}(H_a(x_a), H_b(x_b)) \leq \varepsilon \); 

(iii) for every \( (x_a, x_b) \in R \) and for every \( u_a \in U_a(x_a) \) there exists \( u_b \in U_b(x_b) \) such that for every \( x'_b \in \text{Post}_{u_b}(x_b) \) there exists \( x'_a \in \text{Post}_{u_a}(x_a) \) satisfying 

\[
(x'_a, x'_b) \in R.
\]

We say that \( S_a \) is \( \varepsilon \)-approximately alternatingly simulated by \( S_b \) or that \( S_b \) \( \varepsilon \)-approximately alternatingly simulates \( S_a \), denoted by \( S_a \preceq_{\text{AS}} \varepsilon S_b \), if there exists an \( \varepsilon \)-approximate alternating simulation relation from \( S_a \) to \( S_b \).

Symmetrizing the notion of alternating simulation one obtains the notion of alternating bisimulation.

**Definition 3.2.11** (Approximate alternating bisimulation relation). Let \( S_a \) and \( S_b \) be metric systems with the same output sets \( Y_a = Y_b \) and metric \( \text{d} \), and consider a precision \( \varepsilon \in \mathbb{R}^+ \). A relation \( R \subseteq X_a \times X_b \) is said to be an alternating \( \varepsilon \)-approximate bisimulation relation between \( S_a \) and \( S_b \), if the following two conditions are satisfied:

(i) \( R \) is an alternating \( \varepsilon \)-approximate simulation relation from \( S_a \) to \( S_b \);

(ii) \( R^{-1} \) is an alternating \( \varepsilon \)-approximate simulation relation from \( S_b \) to \( S_a \).

System \( S_a \) is alternating \( \varepsilon \)-approximate bisimilar to \( S_b \), denoted by \( S_a \cong_{\text{AS}} \varepsilon S_b \), if there exists a \( \varepsilon \)-approximate alternating bisimulation relation \( R \) between \( S_a \) and \( S_b \).
Note that for deterministic systems the notion of alternating simulation degenerates into that of simulation. In general, the notions of simulation and alternating simulation are incomparable as illustrated by Example 4.21 in [Tab09]. Also note that for any system \( S_a \), its deterministic counterpart \( S_{d(a)} \) satisfies \( S_a \preceq_{AS} S_{d(a)} \). As in the case of exact simulation relations, we say a 0-approximate alternating simulation relation is an exact alternating simulation relation.

The importance of the preceding notions lies in enabling the transfer of controllers designed for the symbolic models to controllers acting on the original control systems. More details about these notions and how the refinement of controllers is performed can be found in [Tab09].

### 3.2.5 Composition of systems

The feedback composition of a controller \( S_c \) with a plant \( S_a \) describes the concurrent evolution of these two systems subject to synchronization constraints. In this chapter we use the notion of extended alternating simulation relation to describe these constraints. The following formal definition is only used in the proof of Lemma 3.4.4. The readers not interested in the proof can simply replace the symbol \( S_c \times F E S_a \), defined below, with “controller \( S_c \) acting on the plant \( S_a \).

**Definition 3.2.12** (Extended alternating simulation relation [Tab09]). Let \( R \) be an alternating simulation relation from system \( S_a \) to system \( S_b \). The extended alternating simulation relation \( R^c \subseteq X_a \times X_b \times U_a \times U_b \) associated with \( R \) is defined by all the quadruples \( (x_a, x_b, u_a, u_b) \in X_a \times X_b \times U_a \times U_b \) for which the following three conditions hold:

1. \( (x_a, x_b) \in R; \)
2. \( u_a \in U_a(x_a); \)
(iii) \( u_b \in U_b(x_b) \) and for every \( x'_b \in \text{Post}_{u_b}(x_b) \) there exists \( x'_a \in \text{Post}_{u_a}(x_a) \) satisfying \( (x'_a, x'_b) \in R \).

The interested reader is referred to [Tab09] for a detailed explanation on how the following notion of feedback composition guarantees that the behavior of the plant is restricted by controlling only its inputs.

**Definition 3.2.13** (Approximate feedback composition [Tab09]). Let \( S_c \) and \( S_a \) be two metric systems with the same output sets \( Y_c = Y_a \), normed vector spaces, and let \( R \) by an \( \varepsilon \)-approximate alternating simulation relation from \( S_c \) to \( S_a \). The feedback composition of \( S_c \) and \( S_a \) with interconnection relation \( \mathcal{F} = R^\varepsilon \), denoted by \( S_c \times_{\mathcal{F}} S_a \), is the system \((X_{\mathcal{F}}, X_{\mathcal{F}}, U_{\mathcal{F}}, \mathcal{F}, Y_{\mathcal{F}}, H_{\mathcal{F}})\) consisting of:

- \( X_{\mathcal{F}} = \pi_X(\mathcal{F}) = R \);
- \( X_{\mathcal{F}0} = X_{\mathcal{F}} \cap (X_{c0} \times X_{a0}) \);
- \( U_{\mathcal{F}} = U_c \times U_a \);
- \((x_c, x_a) \xrightarrow{(u_c, u_a)}_{\mathcal{F}} (x'_c, x'_a)\) if the following three conditions hold:
  1. \((x_c, u_c, x'_c) \in \xrightarrow{c} ; \)
  2. \((x_a, u_a, x'_a) \in \xrightarrow{a} ; \)
  3. \((x_c, x_a, u_c, u_a) \in \mathcal{F} ; \)
- \( Y_{\mathcal{F}} = Y_c = Y_a \);
- \( H_{\mathcal{F}}(x_c, x_a) = \frac{1}{2}(H(x_c) + H(x_a)) \).

We also denote by \( S_c \times_{\mathcal{F}} S_a \) exact feedback compositions of systems, i.e., whenever \( \mathcal{F} = R^e \) with \( R \) an exact (\( \varepsilon = 0 \)) alternating simulation relation.

### 3.3 Symbolic models for control systems

In this section we show that the time discretization of a \( \delta \)-FC control system admits a countable abstraction model.
3.3.1 Existence of symbolic models

We consider a $\delta$-FC control system $\Sigma = (\mathbb{R}^n, U, U, f)$ and a quadruple $q = (\tau, \eta, \mu, \theta)$ of quantization parameters defining: time quantization $\tau \in \mathbb{R}^+$, state space quantization $\eta \in \mathbb{R}^+$, input space quantization $\mu \in \mathbb{R}^+$, and design parameter $\theta \in \mathbb{R}^+$. For $\Sigma$ and $q$, we define the system:

$$S_q(\Sigma) := (X_q, X_q, U_q, \xrightarrow{q}, Y_q, H_q), \tag{3.6}$$

by:

- $X_q = [\mathbb{R}^n]_\eta$;
- $U_q = [U]_\mu$;
- $x_q \xrightarrow{u_q} x'_q$ if $\|\xi_{x_qu_q}(\tau) - x'_q\| \leq \beta(\theta, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \frac{\eta}{2}.$
- $Y_q = \mathbb{R}^n$;
- $H_q = \iota : X_q \hookrightarrow Y_q.$

The transition relation of $S_q(\Sigma)$ is well defined in the sense that for every $x_q \in X_q$ and every $u_q \in U_q$ there always exists a $x'_q \in X_q$ such that $x_q \xrightarrow{u_q} x'_q$. This can be seen by noting that by definition of $X_q$, for any $x \in \mathbb{R}^n$ there always exists a state $x'_q \in X_q$ such that $\|x - x'_q\| \leq \eta/2$. Hence, for $x = \xi_{x_qu_q}(\tau)$ there always exists $x'_q \in X_q$ satisfying $\|\xi_{x_qu_q}(\tau) - x'_q\| \leq \frac{\eta}{2} \leq \beta(\theta, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \frac{\eta}{2}.$

We remind the reader that the constant $K$ is the Lipschitz constant introduced in the definition of control system used in this chapter.

We stress that while system $S_{\tau}(\Sigma)$ is not countable, system $S_q(\Sigma)$ is so and it becomes finite when the state space of the control system $\Sigma$ is bounded.

We can now state the main result of this section, relating $\delta$-FC to the existence of symbolic models.
Theorem 3.3.1. Let $\Sigma$ be a $\delta$-FC control system. For any desired precision $\varepsilon \in \mathbb{R}^+$, and any quadruple $q = (\tau, \eta, \mu, \theta)$ of quantization parameters satisfying $\eta \leq 2\varepsilon \leq 2\theta$, we have $S_q(\Sigma) \preceq_\varepsilon AS S_\tau(\Sigma) \preceq_\varepsilon S_q(\Sigma)$.

Remark 3.3.2. The transition relation defined in (3.6) can also be written as:

$$x_q \overset{u_q}{\longrightarrow} x'_q \text{ if } B_2(x'_q) \cap B_{\beta(\theta,\tau)+\gamma\left(\frac{\mu + K\tau}{2},\tau\right)}(\xi_{x_q u_q}(\tau)) \neq \emptyset. \quad (3.7)$$

This shows that we place a transition from $x_q$ to any point $x'_q$ for which the ball $B_2(x'_q)$ intersects the over-approximation of $\text{Post}_{u_q}(B_\varepsilon(x_q))$ given by $B_{\beta(\theta,\tau)+\gamma\left(\frac{\mu + K\tau}{2},\tau\right)}(\xi_{x_q u_q}(\tau))$. It is not difficult to see that the conclusion of Theorem 3.3.1 remains valid if we use any over-approximation of the set $\text{Post}_{u_q}(B_\varepsilon(x_q))$.

Theorem 3.3.3. Let $\Sigma$ be a $\delta$-FC control system. For any desired precision $\varepsilon \in \mathbb{R}^+$, and any quadruple $q = (\tau, \eta, \mu, \theta)$ of quantization parameters satisfying $\eta \leq 2\varepsilon \leq 2\theta$, and

$$\beta(\varepsilon, \tau) + 2\gamma\left(\frac{\mu + K\tau}{2}, \tau\right) + \beta(\theta, \tau) + \frac{\eta}{2} \leq \varepsilon, \quad (3.8)$$

we have $S_\tau(\Sigma) \simeq_\varepsilon AS_q(\Sigma)$.

Although the condition $\eta \leq 2\varepsilon$ follows from (3.8), we decided to include it in the statement of Theorem 3.3.3 so that its assumptions can be easily compared with the assumptions in Theorem 3.3.1. Moreover, under a $\delta$-ISS assumption, we can always find a quantization vector $q$ satisfying (3.8).

3.3.2 Simplifications

In Section 3.5 we make use of simpler constructions, resulting from particular choices of $q$. In the case of linear control systems, i.e., $f(x, u) = Ax + Bu$, one can also simplify the theorems just presented in the previous section. These simpler
constructions of abstractions, and particularized theorems for linear systems are collected in the following paragraphs. The proofs of these results are not included in the appendix as they are just special versions of the general theorems for which proofs are provided.

**Definition 3.3.4.** The system

\[ S_{\tau\eta\mu}(\Sigma) = (X_{\tau\eta\mu}, X_{\tau\eta\mu 0}, U_{\tau\eta\mu}, \rightarrow, Y_{\tau\eta\mu}, H_{\tau\eta\mu}) \]

associated with a control system \( \Sigma = (\mathbb{R}^n, U, f) \) and with \( \tau, \eta, \mu \in \mathbb{R}^+ \) consists of:

- \( X_{\tau\eta\mu} = [\mathbb{R}^n]_\eta \);
- \( X_{\tau\eta\mu 0} = X_{\tau\eta\mu} \)
- \( U_{\tau\eta\mu} = \{ v \in U \mid v(t) = v(t') \in [U]_\mu \ \forall t, t' \in [0, \tau] = \text{dom } v \} \);
- \( x \xrightarrow{v_{\tau\eta\mu}} x' \) if there exist \( v \in U \), and a trajectory \( \xi_{xv} : [0, \tau] \to \mathbb{R}^n \) of \( \Sigma \) satisfying \( \|\xi_{xv}(\tau) - x'\| \leq \frac{\eta}{2} \);
- \( Y_{\tau\eta\mu} = \mathbb{R}^n \);
- \( H_{\tau\eta\mu} = : X_{\tau\eta} \hookrightarrow \mathbb{R}^n \).

**Theorem 3.3.5 ([PGT08]).** Let \( \Sigma \) be a linear control system in which all the eigenvalues of the matrix \( A \) have negative real-part and \( U \) consists of piece-wise constant curves. For any desired precision \( \varepsilon \in \mathbb{R}^+ \), time quantization \( \tau \in \mathbb{R}^+ \), input quantization \( \mu \in \mathbb{R}^+ \), and for any space quantization \( \eta \in \mathbb{R}^+ \) satisfying:

\[
\|e^{A\tau}\| \varepsilon + \int_0^\tau \|e^{A t} B\| \, dt \frac{\mu}{2} + \frac{\eta}{2} \leq \varepsilon \tag{3.9}
\]

the following holds:

\[ S_{\tau\eta\mu}(\Sigma) \preceq_{AS} S_\tau(\Sigma). \tag{3.10} \]
We recall now that we denote by $\text{Post}_u(x)$ the set of all the states of $S_\tau(\Sigma)$ that are $u$-successors of $x$. We shall abuse notation and denote by $\text{Post}_u(B_\eta^u(x))$ the set:

$$\text{Post}_u(B_\eta^u(x)) = \bigcup_{x' \in B_\eta^u(x)} \text{Post}_u(x').$$

Using $\text{Post}$ we can construct an abstraction different from the one in Definition 3.3.4.

**Definition 3.3.6.** The system

$$S_{\tau\eta} = (X_{\tau\eta}, X_{\tau\eta}, U_{\tau\eta}, \tau_{\tau\eta}, Y_{\tau\eta}, H_{\tau\eta})$$

associated with a linear control system $\Sigma = (\mathbb{R}^n, U, f)$ and with $\tau, \eta \in \mathbb{R}^+$ consists of:

- $X_{\tau\eta} = [\mathbb{R}^n]_\eta$;
- $X_{\tau\eta,0} = X_{\tau\eta}$
- $U_{\tau\eta} = U$
- $x \xrightarrow{\eta} x'$ if there exist $v \in U$ satisfying:
  \[\text{int}(\text{Post}_v(B_\eta^u(x)) \cap B_\eta^u(x')) \neq \emptyset;\]
- $Y_{\tau\eta} = \mathbb{R}^n$;
- $H_{\tau\eta} = \iota : X_{\tau\eta} \hookrightarrow \mathbb{R}^n$.

Note that the output set $Y_{\tau\eta}$ is naturally equipped with the norm-induced metric $(y, y') = \|y - y'\|$.

Whenever the set $U$ has finite cardinality, the abstract model introduced in Definition 3.3.6 is an abstraction of $S_\tau(\Sigma)$ in the following sense.

**Theorem 3.3.7** ([MT10a]). Let $\Sigma$ be any linear control system with $U$ consisting of piece-wise constant curves and $U$ having finite cardinality. For any desired
precision $\varepsilon \in \mathbb{R}^+$, time quantization $\tau \in \mathbb{R}^+$, and for any space quantization $\eta \in \mathbb{R}^+$ satisfying:

$$\eta \leq 2\varepsilon$$

(3.11)

the following holds:

$$S_{\tau\eta}(\Sigma) \preceq_{AS}^{\varepsilon} S_{\tau}(\Sigma).$$

(3.12)

As shown in [Tab09], and discussed later in this chapter, existence of an approximate alternating simulation relation from $S_{\tau\mu}(\Sigma)$ to $S_{\tau}(\Sigma)$ implies that any controller acting on $S_{\tau\mu}(\Sigma)$ can be refined to a controller acting on $S_{\tau}(\Sigma)$ enforcing the same specification. However, when a controller enforcing the desired specifications on $S_{\tau}(\Sigma)$ exists, there is no guarantee that it can be found by working with the abstraction $S_{\tau\eta}(\Sigma)$. When no controller is found relying on an abstraction $S_{\tau\eta}(\Sigma)$, one needs to search for new parameters $\tau$, $\eta$ and set of inputs $\mathcal{U}$ in order to reattempt the controller synthesis. Therefore, the design flow is not affected by assuming that $\mathcal{U}$ has already been quantized, i.e., that $\mathcal{U}$ has finite cardinality. For this reason, the parameter $\mu$ does not play a role in the assumptions of Theorem 3.3.7.

Remark 3.3.8. The conclusions of Theorem 3.3.7 remain valid if instead of $\text{Post}_u(\mathcal{B}_2(x))$ we use any over-approximation for this set. This is crucial for nonlinear systems and useful for linear systems since over-approximations can be computed much faster than $\text{Post}_u(\mathcal{B}_2(x))$.

3.4 Approximate time-optimal control

In this section we explain how approximate simulation relations can be used to relate time-optimality information, and how making use of this fact one can solve approximately time-optimal control problems in practice.
3.4.1 Problem definition

To simplify the presentation, we consider only systems in which $X_a = Y_a$ and $H_a = 1_{X_a}$. However, all the results in this section can be easily extended to systems with $X_a \neq Y_a$ and $H_a \neq 1_{X_a}$ as we explain at the end of Section 3.4.

**Problem 3.4.1 (Reachability).** Let $S_a$ be a system with $Y_a = X_a$ and $H_a = 1_{X_a}$, and let $W \subseteq X_a$ be a set of states. Let $S_c$ be a controller and $R$ an alternating simulation relation from $S_c$ to $S_a$. The pair $(S_c, F)$, with $F = R^e$, is said to solve the reachability problem if there exists $x_0 \in X_{\mathcal{F}_0}$ such that for every maximal behavior $y \in \mathcal{B}_{x_0}(S_c \times_{\mathcal{F}} S_a) \cup \mathcal{B}^\omega_{x_0}(S_c \times_{\mathcal{F}} S_a)$, there exists $k(x_0) \in \mathbb{N}$ for which $y(k(x_0)) = y_k(x_0) \in W$.

We denote by $\mathcal{R}(S_a, W)$ the set of controller-interconnection pairs $(S_c, F)$ that solve the reachability problem for system $S_a$ with the target set $W$ as specification. For brevity, in what follows we refer to the pairs $(S_c, F)$ simply as controller pairs.

**Definition 3.4.2 (Entry time).** Let $S$ be a system and let $W \subseteq X$ be a subset of states. The entry time of $S$ into $W$ from $x_0 \in X_0$, denoted by $J(S, W, x_0)$, is the minimum $k \in \mathbb{N}$ such that for all maximal behaviors $y \in \mathcal{B}_{x_0}(S) \cup \mathcal{B}^\omega_{x_0}(S)$, there exists some $k' \in [0, k]$ for which $y(k') = y_{k'} \in W$.

If the set $W$ is not reachable from state $x_0$ we define $J(S, W, x_0) = \infty$. Note that asking in Definition 3.4.2 for the minimum $k$ is needed because $S$ might be a non-deterministic system, and thus there might be more than one behavior contained in $\mathcal{B}_{x_0}(S) \cup \mathcal{B}^\omega_{x_0}(S)$ and entering $W$.

If system $S$ is the result of the feedback composition of a system $S_a$ and a controller $S_c$ with interconnection relation $\mathcal{F}$, i.e., $S = S_c \times_{\mathcal{F}} S_a$, we denote by $\tilde{J}(S_c, \mathcal{F}, S_a, W, x_{a0})$ the minimum entry time over all possible initial states of the
controller related to $x_{a0}$:

$$\tilde{J}(S_c, \mathcal{F}, S_a, W, x_{a0}) = \min_{x_{c0} \in X_{c0}} \{J(S_c \times \mathcal{F} S_a, W, (x_{c0}, x_{a0})) \mid (x_{c0}, x_{a0}) \in X_{\mathcal{F}0}\}$$

The time-optimal control problem asks for the selection of the minimal entry time behavior for every $x_0 \in X_0$ for which $J(S, W, x_0)$ is finite.

**Problem 3.4.3 (Time-optimal reachability).** Let $S_a$ be a system with $Y_a = X_a$ and $H_a = 1_{X_a}$, and let $W \subseteq X_a$ be a subset of the set of states of $S_a$. The time-optimal reachability problem asks to find the controller pair $(S_c^*, \mathcal{F}^*) \in \mathcal{R}(S_a, W)$ such that for any other pair $(S_c, \mathcal{F}) \in \mathcal{R}(S_a, W)$ the following is satisfied:

$$\forall x_{a0} \in X_{a0}, \tilde{J}(S_c, \mathcal{F}, S_a, W, x_{a0}) \geq \tilde{J}(S_c^*, \mathcal{F}^*, S_a, W, x_{a0}).$$

### 3.4.2 Entry time bounds

The entry time $J$ acts as the cost function we aim at minimizing by designing an appropriate controller. The following Lemma, which is quite insightful in itself, explains how the existence of an approximate alternating simulation relates the minimal entry times of each system.

**Lemma 3.4.4.** Let $S_a$ and $S_b$ be two systems with $Y_a = X_a$, $H_a = 1_{X_a}$, $Y_b = X_b$ and $H_b = 1_{X_b}$, and let $W_a \subseteq X_a$ and $W_b \subseteq X_b$ be subsets of states. If the following two conditions are satisfied:

- $S_a \preceq_{AS} S_b$ with the relation $R_\epsilon \subseteq X_a \times X_b$;
- $R_\epsilon(W_a) \subseteq W_b$

then the following holds:

$$(x_{a0}, x_{b0}) \in R_\epsilon \implies \tilde{J}(S_{ca}^*, \mathcal{F}_{a}^*, S_a, W_a, x_{a0}) \geq \tilde{J}(S_{cb}^*, \mathcal{F}_{b}^*, S_b, W_b, x_{b0})$$
where \((S^*_c, F^*_c) \in \mathcal{R}(S_a, W_a)\) and \((S^*_b, F^*_b) \in \mathcal{R}(S_b, W_b)\) denote the time-optimal controller pairs for their respective time-optimal control problems, and \(x_{a0} \in X_{a0}, x_{b0} \in X_{b0}\).

We remind the reader now that the proofs of this result and results to follow are compiled in the appendix at the end of this chapter.

The second assumption in Lemma 3.4.4 requires the sets \(W_a\) and \(W_b\) to be related by \(R\). This assumption can always be satisfied by suitably enlarging or shrinking the target sets.

**Definition 3.4.5.** For any relation \(R \subseteq X_a \times X_b\) and any set \(W \subseteq X_b\), the sets \([W]_R, [W]_R^\ast\) are given by:

\[
[W]_R = \{x_a \in X_a \mid R(x_a) \subseteq W\},
\]

\[
[W]_R^\ast = \{x_a \in X_a \mid R(x_a) \cap W \neq \emptyset\}.
\]

The main theoretical result in this section explains how to obtain upper and lower bounds for the optimal entry times in a system \(S_b\) by working with a related system \(S_a\).

**Theorem 3.4.6.** Let \(S_a\) and \(S_b\) be two systems with \(Y_a = X_a\), \(H_a = 1_{X_a}\), \(Y_b = X_b\) and \(H_b = 1_{X_b}\). If \(S_b\) is deterministic and there exists an approximate alternating simulation relation \(R\) from \(S_a\) to \(S_b\) such that \(R^{-1}\) is an approximate simulation relation from \(S_b\) to \(S_a\), i.e.:

\[
S_a \preceq_{AS}^\varepsilon S_b \preceq_{S}^\varepsilon S_a,
\]

then the following holds for any \(W \subseteq X_b\) and \((x_{a0}, x_{b0}) \in R\):

\[
\tilde{J}(S^*_c, F^*_c, S_{d(a)}, [W]_R, x_{a0}) \leq \tilde{J}(S^*_c, F^*_c, S_{b}, W, x_{b0}) \leq \tilde{J}(S^*_c, F^*_c, S_{a}, [W]_R, x_{a0})
\]
where the controller pairs \((S_{cb}^*, F_b^*) \in \mathcal{R}(S_b, W), (S_{ca}^*, F_a^*) \in \mathcal{R}(S_a, [W]_R)\) and
\((S_{cd(a)}^*, F_{d(a)}^*) \in \mathcal{R}(S_{d(a)}, [W]_R)\) are optimal for their respective time-optimal control problems.

**Remark 3.4.7.** If \(S_b\) is not deterministic the inequality
\[
\tilde{J}(S_{cb}^*, F_b^*, S_b, W, x_{b0}) \leq \tilde{J}(S_{ca}^*, F_a^*, S_a, [W]_R, x_{a0})
\]
still holds.

Theorem 3.4.6 explains how upper and lower bounds for the entry times in \(S_b\) can be computed on \(S_a\), hence decoupling the optimality considerations from the specific algorithms used to compute the abstractions. This possibility is of great value when \(S_a\) is a much simpler system than \(S_b\). We exploit this observation in the next section where \(S_b\) denotes a control system and \(S_a\) a much simpler symbolic abstraction.

### 3.4.3 Controller design

Our ultimate objective is to synthesize time-optimal controllers to be implemented on digital platforms. We have shown in Section 3.3 that one can construct, under mild assumptions, symbolic abstractions in the form of finite systems \(S_{abs}\) satisfying \(S_{abs} \preceq S_\tau(\Sigma) \preceq S_{abs}\) with arbitrary precision \(\varepsilon\). Since \(S_{abs}\) is a finite system, entry times for \(S_{abs}\) can be efficiently computed by using algorithms in the spirit of dynamic programming or Dijkstra’s algorithm. It then follows from Theorem 3.4.6 that these entry times immediately provide bounds for the optimal entry time in \(S_\tau(\Sigma)\). Moreover, the process of computing the optimal entry times for \(S_{abs}\) provides us with a time-optimal controller for \(S_{abs}\) that can be refined to an approximately time-optimal controller for \(S_\tau(\Sigma)\). The refined controller is guaranteed to enforce the bounds for the optimal entry times in \(S_\tau(\Sigma)\), computed...
We now present a fixed point algorithm solving the time-optimal reachability problem for finite symbolic abstractions $S_{abs}$. We start by introducing an operator that help us define the time-optimal controller in a more concise way.

**Definition 3.4.8.** For a given system $S_{abs}$ and target set $W \subseteq X_{abs}$, the operator $G_W : 2^{X_{abs}} \rightarrow 2^{X_{abs}}$ is defined by:

$$G_W(Z) = \{ x_{abs} \in X_{abs} \mid x_{abs} \in W \lor \exists u_{abs} \in U_{abs}(x_{abs}) \text{ s.t. } \emptyset \neq \text{Post}_{u_{abs}}(x_{abs}) \subseteq Z \}.$$ 

A set $Z$ is said to be a **fixed point** of $G_W$ if $G_W(Z) = Z$. It is shown in [Tab09] that when $S_{abs}$ is finite, the smallest fixed point $Z$ of $G_W$ exists and can be computed in finitely many steps by iterating $G_W$, i.e., $Z = \lim_{i \to \infty} G_i^W(\emptyset)$. Moreover, the reachability problem admits a solution if the minimal fixed point $Z$ of $G_W$ satisfies $Z \cap X_{abs0} \neq \emptyset$. The time-optimal controller pair can then be constructed from $Z$ as follows:

**Definition 3.4.9 (Time-optimal controller pair).** For any finite system $S_{abs} = (X_{abs}, X_{abs0}, U_{abs}, X_{\rightarrow_{abs}}, 1_{X_{\rightarrow_{abs}}})$ and for any set $W_a \subseteq X_a$, the time-optimal controller pair $(S_{cabs}^*, \mathcal{F}^*) \in R(S_{abs}, W)$ is given by the system $S_{cabs}^* = (X_{cabs}, X_{cabs0}, U_{cabs}, X_{\rightarrow_{cabs}}, 1_{X_{\rightarrow_{cabs}}})$ and by the interconnection relation $\mathcal{F}^* = R^e_{cabs}$ defined by:

- $R_{cabs} = \{(x_{cabs}, x_{abs}) \in X_{cabs} \times X_{abs} \mid x_{cabs} = x_{abs}\}$
- $Z = \lim_{i \to \infty} G_i^W(\emptyset)$;
- $X_{cabs} = Z$;
- $X_{cabs0} = Z \cap X_{abs0}$;
- $x_{cabs} \xrightarrow{u_{abs}}_{cabs} x'_{cabs}$ if there exists a $k \in \mathbb{N}^+$ such that $x_{cabs} \notin G_k^W(\emptyset)$ and $\emptyset \neq \text{Post}_{u_{abs}}(x_{cabs}) \subseteq G_k^W(\emptyset)$,

where $\text{Post}_{u_{abs}}(x_{cabs})$ refers to the $u_{abs}$-successors in $S_{abs}$. 

73
For more details about this controller design we refer the reader to Chapter 6 of [Tab09].

3.4.4 Controller refinement

The time-optimal controller pair \((S^*_{cabs}, \mathcal{F}^*)\) just presented can be easily refined into a controller pair \((S_{cr}(\Sigma), \mathcal{F}_\tau)\) for \(S_\tau(\Sigma)\). Let \(R_{abst}\) be the \(\varepsilon\)-approximate alternating simulation relation from \(S_{abs}\) to \(S_\tau(\Sigma)\), then the refined controller \((S_{cr}(\Sigma), \mathcal{F}_\tau)\) is given by the system \(S_{cr} = (X_{cr}, X_{cr0}, U_{cr}, \xrightarrow{cr}, X_{cr}, 1_{X_{cr}})\) and by the interconnection relation \(\mathcal{F}_\tau = R^c_\tau\) defined by:

- \(R_\tau = \{(x_{cr}, x_\tau) \in X_{cr} \times X_\tau \mid x_{cr} = x_\tau\};\)
- \(X_{cr} = X_\tau;\)
- \(X_{cr0} = X_{\tau0};\)
- \(x_{cr} \xrightarrow{cr} x'_{cr} \) if there exists \(u_{abs} = u_{\tau}, x_{cabs} \in R_{abst}(x_{cr})\) and \(x'_{cabs} \in R^{-1}_{abst}(x'_{cr})\) such that \(x_{cabs} \xrightarrow{cabs} x'_{cabs};\)

where we assumed \(U_{abs} \subseteq U_\tau\).

Intuitively, the refined controller enables all the inputs in \(U_{cabs}(x_{abs})\) at every state \(x_\tau \in X_\tau\) of the system \(S_\tau(\Sigma)\) that is related by \(R_{abst}\) to the state \(x_{abs} \in X_{abs}\) of the abstraction \(S_{abs}\). It is important to notice that this controller is non-deterministic, i.e., at a state \(x_\tau\) all the inputs in \(U_{cr}(x_\tau) = \cup_{x_{abs} \in R^{-1}_{abst}(x_\tau)} U_{cabs}(x_{abs})\) are available and they all enforce the cost bounds.

3.4.5 Approximate time-optimal synthesis in practice

The following is a typical sequence of steps to be followed when applying the presented techniques in practice.
1. **Select a desired precision** $\varepsilon$. This precision is problem dependent and given by practical margins of error.

2. **Construct a symbolic model.** Given $\varepsilon$ construct, using your favorite method, a symbolic model $S_{\text{abs}}$ satisfying: $S_{\text{abs}} \preceq_{\text{AS}}^{\varepsilon} S_{\tau}(\Sigma) \preceq_{\text{S}}^{\varepsilon} S_{\text{abs}}$. Such abstractions can be computed using Pessoa [MDT, MDT10].

3. **Compute the cost’s lower bound.** This bound is obtained as:
   \[
   \tilde{J}(S^*_{\text{cd(abs)}}, \mathcal{F}^*_d, S_{d(abs)}, [W]_R, x_{abs0}) = \min\{k \in \mathbb{N} \mid x_{abs0} \in G^k_{[W]_R}(\emptyset)\} - 1
   \]
   with $G_W$ defined for system $S_{d(abs)}$. This is the best lower bound one can obtain since it follows from Theorem 3.4.4 that by reducing $\varepsilon$ one does not obtain a better lower bound.

4. **Compute the cost’s upper bound.** This bound is obtained as:
   \[
   \tilde{J}(S^*_{\text{abs}}, \mathcal{F}^*, S_{abs}, [W]_R, x_{abs0}) = \min\{k \in \mathbb{N} \mid x_{abs0} \in G^k_{[W]_R}(\emptyset)\} - 1
   \]
   with $G_W$ defined for system $S_{abs}$. The controller obtained when computing this bound, i.e., $S^*_{\text{abs}}$, is the time-optimal controller for $S_{abs}$ and approximately time-optimal for $S_{\tau}(\Sigma)$ after refinement.

5. **Iterate.** If the obtained upper bound is not acceptable, refine the symbolic model so that the new model $S_{\text{abs2}}$ satisfies\(^2\): $S_{\text{abs}} \preceq_{\text{AS}}^{\varepsilon''} S_{\text{abs2}} \preceq_{\text{AS}}^{\varepsilon'} S_{\tau}(\Sigma)$ with $\varepsilon' < \varepsilon$ and $\varepsilon'' < \varepsilon$. In virtue of Theorem 3.4.4 (and Remark 3.4.7) the upper bound will not increase. Moreover, it is our experience that, in general, the upper bound will improve by using more accurate symbolic models, i.e., $\varepsilon' < \varepsilon$.

The more general case where $X_\tau \neq Y_\tau$, $H_\tau \neq 1_{X_\tau}$ and one is given an output target set $W_Y \subseteq Y$ can be solved in the same manner by using the target set $W \subseteq X$ defined by $W = H^{-1}(W_Y)$.

\(^2\)The constructions in Section 3.3 satisfy this property with $\varepsilon = \eta/2$, $\varepsilon' = \eta'/2$ and $\varepsilon'' = \frac{\eta - \eta'}{2}$ by selecting $\eta' = \frac{\eta}{2}$ with $\rho > 1$ an odd number and $\theta = \varepsilon$, $\theta' = \varepsilon'$. 

75
3.5 **Pessoa: A Matlab toolbox for the synthesis of correct-by-design embedded controllers**

**Pessoa** is a toolbox automating the synthesis of correct-by-design embedded control software. Although the core algorithms in **Pessoa** have been coded in C, the main functionalities are available through the **Matlab** command line. **Pessoa** Version 1.0 offers three main functionalities:

1. the construction of finite symbolic models of linear\(^3\) control systems;
2. the synthesis of symbolic controllers for simple specifications;
3. simulation of the closed-loop behavior in **Simulink**.

Each one of these functionalities is described in more detail in the following sections.

### 3.5.1 Binary Decision Diagrams

All the systems and sets manipulated by the toolbox **Pessoa**, described in Section 3.5, are represented symbolically using Reduced Ordered Binary Decision Diagrams (ROBDDs) supported by the CUDD library [CUD]. Binary Decision Diagrams (BDDs) are efficient data structures used to store boolean functions. A BDD, also known as a branching program, is a directed acyclic graph in which all nodes have out degree two, except for the output nodes. Intuitively, it is a tree with as many levels as bits in the domain of the boolean function \(\delta_a\) to be represented. The tree has one start node and two final leaves (the output nodes).

\(^3\)Linear control systems are natively supported in **Pessoa** Version 1.0. Nonlinear and switched systems can also be handled by **Pessoa** but require some additional effort by the user. For further information please consult the documentation in [http://www.cyphylab.ee.ucla.edu/Pessoa](http://www.cyphylab.ee.ucla.edu/Pessoa).
labeled *true* and *false*, representing the output of $\delta_a$. To evaluate the function $\delta_a$, one proceeds to follow the tree from its root. At each level $i$ a branch is selected depending on the value of the $i$-th bit of the input to $\delta_a$ until a final leaf is reached, which provides the value of $\delta_a$ under the given input. BDD representations exhibit many advantages for verification purposes [Weg00]. We remark their effective use of space when using their canonical form: *Reduced Ordered BDD (ROBDD)* [Weg00]. We employ BDDs to represent finite systems by transforming the transition relation into a boolean function. If for a given system $S_a$ we know that the cardinalities of $X_a$ and $U_a$ are $|X_a| \leq 2^{n_x}$ and $|U_a| \leq 2^{n_u}$, the transition relation $\xrightarrow{a}$ admits the alternative representation as a Boolean function $\delta_a : \mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \times \mathbb{B}^{n_x} \rightarrow \mathbb{B}$, where:

$$\delta_a(b_{n_x}(x), b_{n_u}(u), b_{n_x}(x')) = true \Leftrightarrow (x, u, x') \in \xrightarrow{a}.$$ 

3.5.2 Software design as a controller synthesis problem

Regarding software design as a controller synthesis problem is an idea that has been recently gaining enthusiasts despite having been proposed more than 20 years ago [EC82, MW84]. The starting point is to regard the software to be designed as a system $S_{\text{cont}}$ such that the composition $S_{\text{cont}} \times S_{\tau}(\Sigma)$ satisfies the desired specification. If the specification is given as another system $S_{\text{spec}}$, then we seek to synthesize a controller $S_{\text{cont}}$ so that:

$$S_{\text{cont}} \times S_{\tau}(\Sigma) \preceq S_{\text{spec}},$$

or even:

$$S_{\text{cont}} \times S_{\tau}(\Sigma) \cong S_{\text{spec}}.$$

In general, this problem is not solvable algorithmically since $S_{\tau}(\Sigma)$ is an infinite system. We overcome this difficulty by replacing $S_{\tau}(\Sigma)$ by a finite abstraction.
$S_{abs}$ for which we have the guarantee that if a controller satisfying:

$$S_{cont} \times S_{abs} \preceq_S S_{spec}$$

exists then a controller $S'_{cont}$ satisfying:

$$S'_{cont} \times S_{\tau}(\Sigma) \preceq_S S_{spec}$$

also exists. We call $S'_{cont}$ the refinement of $S_{cont}$. It is shown in [Tab09] that existence of an approximate alternating simulation relation from $S_{abs}$ to $S_{\tau}(\Sigma)$ is sufficient to refine the controller $S_{cont}$ acting on $S_{abs}$ to the controller $S'_{cont}$ acting on $S_{\tau}(\Sigma)$. If we can also establish the existence of an approximate alternating bisimulation relation between $S_{abs}$ and $S_{\tau}(\Sigma)$, then we have the guarantee that if a controller exists for $S_{\tau}(\Sigma)$, a controller also exists for $S_{abs}$. Hence, this design flow is not only sound but also complete. Moreover, since $S'_{cont}$ admits a finite description, it can be directly compiled into code executable in a digital platform.

### 3.5.3 Computing symbolic models in Pessoa

We discuss in this section some practical issues regarding the construction of symbolic models in Pessoa. We concentrate on the construction of abstractions for linear systems, which are natively supported in Pessoa. Linearity of the control system being abstracted is exploited by Pessoa in different ways to simplify the computations. In particular, we make use of the variation of constants formula, i.e., given a state $x \in X$ and a constant input $v \in U$, the $v$-successor of $x$ in $S_{\tau}(\Sigma)$, given by $\xi_{xv}(\tau)$, can be computed as:

$$\xi_{xv}(\tau) = e^{A\tau}x + \int_0^\tau e^{A(\tau-t)} Bv(t)dt.$$ 

We can thus express $\text{Post}_v(B_{\frac{\tau}{2}}(x))$ as:

$$\text{Post}_v(B_{\frac{\tau}{2}}(x)) = A_\tau(B_{\frac{\tau}{2}}(x)) \oplus \{B_{\tau}v\}$$
where the matrices $A_\tau$ and $B_\tau$ are defined by:

$$A_\tau = e^{A\tau}, \quad B_\tau = \int_0^\tau e^{A(\tau-t)}Bdt.$$ 

The closed ball $B_2(x)$ can be written as:

$$B_2(x) = \{x\} \oplus B_2(0)$$

and leads to the decomposition:

$$\text{Post}_v(B_2(x)) = \{A_\tau x\} \oplus A_\tau(B_2(0)) \oplus \{B_\tau v\}.$$ 

Note that the second and third terms can be computed only once, when evaluating $\text{Post}_u(B_2(x))$ at the states $x \in X_\eta$, since they do not depend on $x$. To speedup the computations further, the set $A_\tau(B_2(0))$ is not computed exactly, but rather over-approximated as a union of hyper-rectangles commensurable with $\eta$. Despite this approximation, we still obtain an abstraction satisfying (3.3.7) as explained in Remark 3.3.8. The abstraction $S_{r\mu}(\Sigma)$ introduced in Definition 3.3.4 does not require over-approximations since $\xi_{xu}(x)$ is readily computed as $A_\tau x + B_\tau v$.

The transition relations $\longrightarrow$ of $S_{r\mu}(\Sigma)$ and $S_{r\eta}(\Sigma)$ are encoded in a ROBDD through the corresponding characteristic functions, i.e., we encode the binary function:

$$T : X \times U \times X \rightarrow \{0, 1\}$$

satisfying $T(x, u, x') = 1$ iff $(x, u, x') \in \longrightarrow$. To speed up the computation of the ROBDD describing the function $T$, we first perform a change of coordinates taking $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ to $X \subseteq \mathbb{Z}^n$ and $U \subseteq \mathbb{Z}^m$. In this manner we use the unsigned integer variables to encode the states and inputs, and to perform all the computations.

Finally a comment on the finiteness of the abstractions appears necessary. The symbolic models in Section 3.3 have countably infinite state sets. However,
in practical applications the physical variables are restricted to a compact set. Velocities, temperatures, pressures, and other physical quantities cannot become arbitrarily large without violating the operational envelop defined by the control problem being solved. By making use of this fact, $S_{\tau \eta \mu}(\Sigma)$ and $S_{\tau \eta}(\Sigma)$ can be regarded as finite systems. To simplify the discussion in this paragraph, we will use $S_\bullet(\Sigma) = (X_\bullet, X_\bullet 0, U_\bullet, \rightarrow, Y_\bullet, H_\bullet)$ to refer to both $S_{\tau \eta \mu}(\Sigma)$ and $S_{\tau \eta}(\Sigma)$. The first observation is that we can encode the operational envelop on the output map of $S_\bullet(\Sigma)$. We thus consider a compact set $D \subset \mathbb{R}^n$ and redefine the output set of $S_\bullet(\Sigma)$ to $Y_\bullet = D \cup \{*\}$ for some element * not belonging to $D$. The symbol * represents all the states that are “out of bounds” or “out of sensor range”. The output map of $S_\bullet(\Sigma)$ is also redefined to:

$$H_\bullet(x) = \begin{cases} x & \text{if } x \in X \cap D \\ * & \text{if } x \notin X \cap D \end{cases}$$

The new output set is equipped with the metric:

$$(y, y') = \begin{cases} \frac{1}{2} \text{diam}(D) & \text{if } y' = * \text{ and } y \in D \\ 0 & \text{if } y = * = y' \\ \|y - y'\| & \text{if } y, y' \in D \end{cases}$$

Although the redefined system $S_\bullet(\Sigma)$ is still countably infinite, it $0$-approximately alternatingly simulates the finite system $S_{abs} = (X_{abs}, X_{abs 0}, U_{abs}, \rightarrow_{abs}, Y_{abs}, H_{abs})$ consisting of:

- $X_{abs} = [D]\eta \cup \{*\}$;
- $X_{abs 0} = X_{abs} \cap X_\bullet 0$;
- $U_{abs} = U_\bullet$;
- $x \rightarrow_{abs} x'$ in $S_{abs}$ if $x, x' \in [D]\eta$ and $x \rightarrow x'$ in $S_\bullet(\Sigma)$ or if $x \in [D]\eta$, $x' = *$, and $x \rightarrow x''$ in $S_\bullet(\Sigma)$ with $x'' \in X_\bullet \setminus [D]\eta$;
\[ Y_{abs} = Y_\bullet; \]
\[ H_{abs} = 1_{X_{abs}}. \]

The relation \( R \subseteq X_{abs} \times X_\bullet \) defined by \( (x_{abs}, x_\bullet) \in R \) if \( x_{abs} = x_\bullet \in [D]_\eta \) or \( x_{abs} = * \) and \( x' \in X \setminus [D]_\eta \) is a 0-approximate alternating simulation relation from \( S_{abs} \) to \( S_\bullet(\Sigma) \). Finiteness of \( S_{abs} \) now follows from compactness of \( D \). Intuitively, \( S_{abs} \) is not more than the restriction of \( S_\bullet(\Sigma) \) to the set \( D \). For this reason, we implicitly assume that all the specifications that we are interested in contain the requirement that no trajectory should ever leave the set \( D \), even if this is not explicitly stated.

3.5.4 Synthesizing symbolic controllers in Pessoa

Pessoa currently supports the synthesis of controllers enforcing four\(^4\) kinds of specifications defined using a target set \( Z \subseteq X \) and a constraint set \( W \subseteq X \):

1. **Stay**: trajectories start in the target set \( Z \) and remain in \( Z \). This specification corresponds to the Linear Temporal Logic (LTL) formula\(^5\) \( \square \varphi_Z \) where \( \varphi_Z \) is the predicate defining the set \( Z \);
2. **Reach**: trajectories enter the target set \( Z \) in finite time. This specification corresponds to the LTL formula \( \Diamond \varphi_Z \);
3. **Reach and Stay**: trajectories enter the target set \( Z \) in finite time and remain within \( Z \) thereafter. This specification corresponds to the LTL formula \( \Diamond \square \varphi_Z \);
4. **Reach and Stay while Stay**: trajectories enter the target set \( Z \) in finite time and remain within \( Z \) thereafter while always remaining within the

\(^{4}\) Future versions of Pessoa will handle specifications given as linear temporal logic formulas or automata on infinite strings.

\(^{5}\) The semantics of LTL would be defined in the usual manner over the output behaviors of \( S_\tau(\Sigma) \).
constraint set $W$. This specification corresponds to the LTL formula
\[ \Diamond \Box \varphi_Z \land \Box \varphi_W \]
where $\varphi_W$ is the predicate defining the set $W$.

Although simple, the above specifications already allow Pessoa to solve non-trivial synthesis problems as described in Section 3.6. Reach and stay specifications can be used to encode usual set regulation problems where the state is to be steered to a desired operating point set and forced to remain there. The fourth kind of specification complements reach and stay requirements by imposing state constraints, defined by the set $W$, that are to be enforced for all time.

The controllers for the above specifications are memoryless controllers that can be synthesized through fixed point computations as described in [Tab09]. All the fixed-points are computed symbolically using the ROBDD representation of the abstractions $S_{\tau_\mu}(\Sigma)$ or $S_{\tau_\eta}(\Sigma)$, and a ROBDD representation for the sets $Z$ and $W$. These sets can be specified as hyper-rectangles, by providing the corresponding vertices, or as arbitrary sets, by providing the corresponding characteristic functions. The finite state nature of the synthesized controllers permits a direct compilation into code. Although code generation is not yet supported in Version 1.0 of Pessoa, closed-loop simulation in Simulink is already available.

3.5.5 Simulating the closed-loop in Simulink

Pessoa also provides the possibility to simulate the closed-loop behavior in Simulink. For this purpose, Pessoa comes with a Simulink block implementing a refinement of any synthesized controller (see Figure 3.2). The controllers synthesized in Pessoa are, in general, nondeterministic. The Simulink block resolves this non-determinism in a consistent fashion thus providing repeatable simulations. In order to increase the simulation speed, the Simulink block selects, among all the
inputs available for the current state, the input with the shortest description in the ROBDD encoding the controller. Moreover, the input is chosen in a lazy manner, i.e., the input is only changed when the previously used input cannot be used again. Other determinization strategies, such as minimum energy inputs, will be supported in future versions of Pessoa.

3.6 Examples

We provide in this section examples illustrating the power of the techniques presented. The following examples were all implemented on Pessoa. All the run-time values for the examples were obtained on a MacBook with 2.2 GHz Intel Core 2 Duo processor and 4GB of RAM.

3.6.1 DC Motor

The first example can be found in most undergraduate control textbooks and consists in regulating the velocity of a DC motor. The electric circuit driving the DC motor is shown in Figure 3.1. The dynamics $\Sigma$ of this system comprises two

![Figure 3.1: DC motor and associated electric circuit.](image)

The first example can be found in most undergraduate control textbooks and consists in regulating the velocity of a DC motor. The electric circuit driving the DC motor is shown in Figure 3.1. The dynamics $\Sigma$ of this system comprises two
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$500 \times 10^{-3}$</td>
<td>Resistance</td>
</tr>
<tr>
<td>L</td>
<td>$1500 \times 10^{-6}$</td>
<td>Inductance</td>
</tr>
<tr>
<td>J</td>
<td>$250 \times 10^{-6}$</td>
<td>Moment of inertia</td>
</tr>
<tr>
<td>B</td>
<td>$100 \times 10^{-6}$</td>
<td>Viscous friction coefficient</td>
</tr>
<tr>
<td>k</td>
<td>$50 \times 10^{-3}$</td>
<td>Torque constant</td>
</tr>
</tbody>
</table>

Table 3.1: Parameters for the circuit in Figure 3.1 expressed in the international system of units.

The linear differential equations:

\[
\dot{x}_1 = -\frac{B}{J} x_1 + \frac{k}{J} x_2 \tag{3.13}
\]

\[
\dot{x}_2 = -\frac{k}{L} x_1 - \frac{R}{L} x_2 + \frac{1}{L} u. \tag{3.14}
\]

The variable $x_1$ describes the angular velocity of the motor, the variable $x_2$ describes the current $i$ through the inductor, and the variable $u$ represents the source voltage $v$ that is treated as an input. The model parameters are shown in Table 3.1.

The control objective is to regulate the velocity around 20 rad/s. We select the domain $D$ for the symbolic model to be:

\[ D = [-1, 30] \times [-10, 10]. \]

The input space is $U = [-10, 10]$ and the quantization parameters are given by $\tau = 0.05$, $\eta = 0.5$, and $\mu = 0.01$. These quantization parameters were chosen so as to satisfy inequality (3.9) in Theorem 3.3.5 with $\varepsilon = 1$. Since the objective is to regulate the velocity to a desired set point, we consider the target set:

\[ Z = [19.5, 20.5] \times [-10, 10] \]

84
constraining the velocity to a neighborhood of the desired set point and chose a “reach and stay” specification in Pessoa. The symbolic abstraction was computed in 18 minutes while the symbolic controller took less than one second to be synthesized. The closed loop behavior was simulated in Simulink using the symbolic controller block included in Pessoa and represented in Figure 3.2. The evolution of the velocity and input are displayed in Figure 3.3 for the initial condition \((x_1, x_2) = (0, 0)\).

In practical implementations the DC motor is connected to a constant voltage source through an H-bridge. By opening and closing the switches in the H-bridge we can only chose three different values for the voltage: \(-10\text{V}, 0\text{V}, \text{ and } 10\text{V}\). In order to synthesize a controller under these input constraints we redefine the input quantization to \(\mu = 10\). This guarantees that \(u\) can only assume the desired three voltage levels. Velocity regulation now requires more frequent changes to the input voltage. Hence, we change the time quantization to \(\tau = 0.0001\) and also the space quantization \(\eta = 0.05\) so that we can capture the changes that occur during each sampling period of 0.0001 seconds. These quantization parameters no longer satisfy inequality (3.9) and settle for a symbolic abstraction related to

Figure 3.2: Simulink diagram for the closed-loop system depicting the symbolic controller block included in Pessoa.
$S_τ(\Sigma)$ by an approximate alternating simulation. The abstraction is computed in 17 minutes and the controller synthesized in 108 seconds.

The time evolution of the velocity and current are obtained by simulating the closed-loop system with the new controller and can be seen in Figure 3.4. Although the velocity converges to a small neighborhood of 20 rad/s (see Figure 3.5), the values of the current through the inductor are quite large, attaining a peak of 10 Amperes. This can be improved by redefining the target set to:

$$Z = [19.5, 20.5] \times [-0.7, 0.7]$$

so as to reduce the current ripple to 0.7 Amperes around 0, and by introducing the constraint set $W$:

$$W = [-1, 30] \times [-3, 3]$$

to limit the peak current to 3 Amperes. We synthesize a new controller enforcing the “reach and stay while stay” in 88 seconds. The closed-loop simulation results in Figure 3.6 show that the target set is still reached while the current ripple and peak values have been reduced to conform to the new target set and constraint set. Note how the peak current limitation forces slows the convergence to the target set $Z$.

### 3.6.2 Control with shared actuators

The second example addresses the problem of controller synthesis under shared resources. We consider a control system that has permanent access to a low quality actuator and sporadic access to a high quality actuator. This scenario arises when the high quality actuator is connected to the controller through a shared network, or consumes large amounts of energy drawn from a shared battery. Moreover, we also assume that we do not have at our disposal a model for the
other software tasks competing for the shared resources. This is typically the case when such software tasks are being concurrently designed. However, even if we had models for these software tasks, the complexity of synthesizing the control software using these models would be prohibitive. Therefore, we shall impose a simple fairness requirement mediating the access to the shared resources.

To make the ensuing discussion concrete, we assume that three tasks can have access to the shared resources, one of them being the control task. We use the expression time slot to refer to time intervals of the form $[k\tau, (k+1)\tau]$ with
Figure 3.5: Evolution of velocity and input when the input voltage is restricted to $-10$, $0$, and $10$ Volts.

$k \in \mathbb{N}$ and where $\tau$ is the time quantization parameter. If we consider sequences of three consecutive time slots, the fairness requirement imposes the availability of the actuator in at least one time slot. Possible availability sequences satisfying this assumption are:

|aaa|aaa|aaa|aaa|aaa|aaa|aaa|aaa|aaa|...
|aua|uaa|aua|aua|aua|aua|aua|aua|aua|...
|aau|aau|aau|aau|aau|aau|aau|aau|aau|...
|uaa|uau|uua|uua|uua|uua|uua|uua|uua|...
|uaa|uau|uua|uua|uua|uua|uua|uua|uua|...
|uau|uau|uau|uau|uau|uau|uau|uau|uau|...

where we denoted by $a$ the availability of the resources, by $u$ the unavailability, and separated the sequences of three time slots with the symbol $|$. Since the preceding sequences form an $\omega$-regular language they can be described by the automaton represented in Figure 3.7. The system $\Sigma$ to be controlled is a double
Figure 3.6: Evolution of velocity and current when the input voltage is restricted to $-10$, 0, and 10 Volts and state constraints are enforced.

![Figure 3.6](image)

Figure 3.7: Automaton describing the availability of the shared resources. The lower part of the states is labeled with the outputs $a$ and $u$ denoting availability and unavailability of the shared resource, respectively.

integrator:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= u_{\text{low}} + u_{\text{high}}.
\end{align*}
\]

where $u_{\text{low}}$ denotes the input produced by the low quality actuator and $u_{\text{high}}$ denotes the input produced by the high quality actuator. Any of the actuators generates piecewise constant inputs taking values in $U = [-1, 1]$. However, when an input $u \in U$ is requested to the low quality actuator, the actual generated input $u_{\text{low}}$ is an element of the set $[u - 0.6, u + 0.6]$. In contrast, the high quality actuator always produces the input that is requested, i.e., $u_{\text{high}} = u$. 

The control objective is to force the trajectories to remain within the target set \( Z = [-1, 1] \times [-1, 1] \). The fairness constraint is also a control objective that can be expressed by resorting to a model for the concurrent execution of \( S_\tau(\Sigma) \) and the automaton in Figure 3.7. When the automaton is in state \( q_1 \), any of the actuators can be used. However, when the automaton is in the state \( q_2 \) or \( q_3 \) only the low quality actuator can be used. Although this kind of specification is not natively supported in Pessoa, it can be handled by providing Pessoa with a Matlab file containing an operational model for the concurrent execution of \( S_\tau(\Sigma) \) and the automaton in Figure 3.7. Choosing \( D = [-1, 1] \times [-1, 1] \) as the domain of the symbolic abstraction, and \( \tau = 0.1 \), \( \eta = 0.05 \), and \( \mu = 0.5 \) as quantization parameters, Pessoa computes the symbolic abstraction in 109 seconds and synthesizes a controller in 2 seconds. The domain of the controller is shown in Figure 3.8 and two typical closed-loop behaviors are shown in Figures 3.9, 3.10, and 3.11. We can appreciate the controller forcing the trajectories to stay within the target set despite the low quality of the permanently available actuator. We note that if we require the high quality actuator to be permanently unavailable, Pessoa reports the non-existence of a solution.

### 3.6.3 Approximate time-optimal control of a double integrator

We illustrate the approximate time-optimal control technique on the classical example of the double integrator, where \( \Sigma \) is the control system: 

\[
\dot{\xi}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \xi(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(t)
\]

and the target set \( W \) is the origin, i.e., \( W = \{(0, 0)\} \).

Following the steps presented in Section 3.4, first we select a precision \( \varepsilon \) which in this example we take as \( \varepsilon = 0.15 \). Next, we relax the problem by enlarging the
Figure 3.8: Domain of the controller forcing the double integrator to remain in $[-1, 1] \times [-1, 1]$ under the fairness constraints described by the automaton in Figure 3.7.

target set to $W = B_1((0, 0))$. We select as parameters for the symbolic abstraction $\tau = 1, \mu = 0.1$ and $\eta = 0.3$. Restricting the state set to $X = B_{30}((0, 0)) \subset \mathbb{R}^2$ the state set of $S_\tau(\Sigma)$ becomes finite and the proposed algorithms can be applied. Constructing the abstraction $S_\tau(\Sigma)$ in Pessoa took less than 5 minutes and the resulting model required 7.9 MB to be stored. The lower bound required about 50 milliseconds while computing the time-optimal controller required only 3 seconds and the controller was stored in 1 MB.

The approximately time-optimal controller $S^*_c$ is depicted in Figure 3.12(a). We remind the reader that the obtained controller is non-deterministic. Hence, Figure 3.12(a) shows one of the valid inputs of the time-optimal controller at different locations of the state-space. The optimal controller to the origin is also shown in Figure 3.12(a) represented by the switching curve (thick blue line) dividing the state space into regions where the inputs $u = 1$ (below the switching curve) and $u = -1$ (above the switching curve) are to be used. As expected, the partition produced by this switching curve does not coincide with the one
Figure 3.9: Evolution of the state variables (left figure) and inputs (right figure), from initial state \((x_1, x_2) = (-1, 0.8)\), when the automaton in Figure 3.7 is visiting the states \(|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|\ldots\). The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

found by our toolbox, as the time-optimal controller reported in [PBG62] is not time-optimal to reach the set \(W\) (it is just optimal when the target set is the singleton \(\{(0,0)\}\)).

Although the computed bounds are conservative, the cost achieved with the symbolic controller is quite close to the true optimal cost as illustrated in Figure 3.12(b) and Table 3.2. This is a consequence of the bounds relying entirely on the worst case scenarios induced by the non-determinism of the computed abstractions. In practice, the symbolic controller determines the actual state of the system every time it acquires a state measurement thus resolving the non-determinism present in the abstraction. In Figure 3.12(b) we present the ratio between the cost to reach \(W\), obtained from the symbolic controller, and the time-optimal controller. The time-optimal controller to reach the origin operates in continuous time and thus for some regions of the state-space the cost obtained will be smaller than one unit of time. On the other hand, the approximate time-
Figure 3.10: Evolution of the state variables (left figure) and inputs (right figure), from initial state \((x_1, x_2) = (-1, 0.8)\), when the automaton in Figure 3.7 is visiting the states \(q_1q_2q_1|q_2q_1q_2|q_1q_2q_1|q_2q_1q_2|q_1q_2q_1|\ldots\). The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

optimal controller obtained with our techniques cannot obtain costs smaller than one unit of time, as it operates in discrete time. Hence, to make the comparison fair, in Figure 3.12(b) the costs achieved by the time-optimal controller smaller than one unit of time were saturated to a cost of 1 time unit. In Table 3.2 specific values of the time to reach the target set \(W\) using the constructed controller are compared to the cost of reaching \(W\) with the true time-optimal controller to reach the origin.

3.6.4 Approximate time-optimal control of a unicycle

Finally, we want to persuade the reader with this example of the potential of the presented techniques to solve control problems with both qualitative and quantitative specifications. The problem we consider now is to drive a unicycle through a given environment with obstacles. In this example both qualitative and quantita-
Figure 3.11: Evolution of the state variables. The left figure refers to the initial states and automaton evolution in Figure 3.9 while the right figure refers to the initial states and automaton evolution in Figure 3.10.

<table>
<thead>
<tr>
<th>Initial State</th>
<th>$(-6.1, 6.1)$</th>
<th>$(-6, 6)$</th>
<th>$(-5.85, 5.85)$</th>
<th>$(3.1, 0.1)$</th>
<th>$(3.0)$</th>
<th>$(2.85, -0.1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>12.83 s</td>
<td>12.66 s</td>
<td>11.60 s</td>
<td>2.66 s</td>
<td>2.53 s</td>
<td>2.38 s</td>
</tr>
<tr>
<td>Symbolic</td>
<td>14 s</td>
<td>14 s</td>
<td>13 s</td>
<td>3 s</td>
<td>3 s</td>
<td>3 s</td>
</tr>
<tr>
<td>UpperBound</td>
<td>29 s</td>
<td>29 s</td>
<td>29 s</td>
<td>7 s</td>
<td>7 s</td>
<td>7 s</td>
</tr>
<tr>
<td>LowerBound</td>
<td>9 s</td>
<td>9 s</td>
<td>9 s</td>
<td>2 s</td>
<td>2 s</td>
<td>2 s</td>
</tr>
</tbody>
</table>

Table 3.2: Times achieved in simulations by a time-optimal controller to reach the origin and the symbolic controller.

tive specifications are provided. The avoidance of obstacles prescribes conditions that the trajectories should respect, thus establishing qualitative requirements of the desired trajectories. Simultaneously, a time-optimal control problem is specified by requiring the target set to be reached in minimum time, thus defining the quantitative requirements. Hence, the complete specification requires the synthesis of a controller disabling trajectories that hit the obstacles, and selecting, among the remaining trajectories, those with the minimum time-cost associated to them.
Figure 3.12: (a) Symbolic controller $S_c^*$. (b) Time to reach the target set $W$ represented as the ratio between the times obtained from the symbolic controller and the times obtained from the continuous time-optimal controller to reach the origin.

We consider the following model for the unicycle control system:

$$\dot{x} = v \cos(\theta), \quad \dot{y} = v \sin(\theta), \quad \dot{\theta} = \omega$$

in which $(x, y)$ denotes the position coordinates of the vehicle, $\theta$ denotes its orientation, and $(v, \omega)$ are the control inputs, linear velocity and angular velocity respectively. The parameters used in the construction of the symbolic model are: $\eta = 0.2$, $\mu = 0.1$, $\tau = 0.5$ seconds, and $v \in [0, 0.5]$ and $\omega \in [-0.5, 0.5]$. The problem to be solved is to find a feedback controller optimally navigating the unicycle from any initial position to the target set $W = [4.6, 5] \times [1, 1.6] \times [-\pi, \pi]$, indicated with a red box in Figure 3.13 (with any orientation $\theta$), while avoiding the obstacles in the environment, indicated as blue boxes in Figure 3.13. The symbolic model was constructed in 179 seconds and used 11.5 MB of storage, and the approximately time-optimal controller was obtained in 5 seconds and required 3.5 MB of storage. In Figure 3.13 we present the result of ap-
plying the approximately time-optimal controller with the prescribed qualitative requirements (obstacle avoidance). The (approximately) bang-bang nature of the obtained controller can be appreciated in the right plot of this figure. For the initial condition \((1.5, 1, 0)\) the solution obtained, presented in Figure 3.13, required 44 seconds to reach the target set.

Figure 3.13: Unicycle trajectory under the automatically generated approximately time-optimal feedback controller (left figure) and the inputs employed: \(v\) in yellow and \(\omega\) in pink (right figure).

### 3.7 Discussion

In this chapter we have presented symbolic abstractions for control systems approximately alternatingly simulated by the discrete time version of the original control system. Approximate alternating simulations allows to refine controllers for the symbolic abstractions into controllers for the original systems. These relations can be strengthen into approximately alternating bisimulation relations when the original control system is incrementally input-to-state stable. Bisimulations are desirable as, if a controller for the original system satisfying the specification exists, a controller for the symbolic model also exists. On the other
hand, if only an approximately alternating simulation relation is obtained, failing to find a controller on the symbolic model does not prevent the existence of a controller for the original control system. In view of this observation, control designers are now asked to choose between the following two alternatives:

1. design a controller rendering the original control system incrementally input-to-state stable and then apply the abstraction techniques providing approximate alternating bisimulation relations [PGT08, PT09, Gir07, GPT09];
2. or construct an abstraction for the original system (under the assumption of incremental forward completeness), and risk not finding a controller for such abstraction.

The following result, proved in [ZPM10], addresses this question: existence of a controller rendering the original control system incrementally input-to-state stable implies that, if a controller can be found using the abstraction for the incrementally input-to-state stable system then, a controller enforcing the same specification can be found using the abstractions in Section 3.3 for the original system. Yet, following the first alternative simplifies the selection of quantization parameters in the proposed abstractions to find solutions to the specified problem. The development of techniques for the design of controllers rendering the closed-loop system incrementally stable is already being investigated [ZT10].

In order to make all these theoretical advances truly practical, the size of the synthesized controllers needs to be optimized. This can be achieved by exploiting symmetries and other structure of the already synthesized controllers or by reducing the size of the abstractions employed in the synthesis. The abstractions proposed in Section 3.3 suffer from the curse of dimensionality due to the grids employed on state and inputs sets. The state set of the abstractions in Section 3.3 is a grid of resolution $\eta$. However, Theorem 3.3.7 does not require the use of a
grid of constant resolution. Thus, the use of multi-resolution grids can help reducing the size of the computed abstractions and controllers. Abstractions based in such grids are, to a certain degree, still applicable to the solution of different specifications. However, in general, the selection of the different resolutions would be greatly influenced by the targeted problem. The size of the abstractions can be further reduced by adapting them completely to the particular specification given, e.g. by relying on partitions of the state set adapted to the sets involved in the specification. Both of these approaches are subject of future study and functionalities to be included in future releases of Pessoa.

We have also addressed the suitability of symbolic abstractions to the resolution of problems with both qualitative and quantitative specifications. The quantitative specifications we addressed are given in the form of time-optimal reachability problems. We have shown that information about time-optimality can be inferred from approximate alternating simulation relations between systems. Further research is needed to formalize similar statements for more general optimal control problems. Other quantitative specifications, with a great impact in real applications, are those including levels of robustness. Solutions to such problems involve the quantification of the relative degradation of the system behavior with respect to quantified disturbances and/or modeling inaccuracies.

On the more practical side, we have introduced a tool developed in support of the presented theories named Pessoa. Besides the aforementioned extensions to Pessoa other more practical ones are already under development. Specifications with discrete memory can be used with Pessoa by encoding them in the plant dynamics as reported in Section 3.6.2. Also, nonlinear and switched dynamics can already be used in Pessoa, albeit not natively. Pessoa is currently being extended to natively support specifications given in LTL and automata on infinite strings,
and provide native support for non-linear and switched systems.

3.8 Appendix: Proofs

The proof of Theorem 3.3.1 requires the following technical Lemma.

**Lemma 3.8.1.** Let $\Sigma = (\mathbb{R}^n, U, \mathcal{U}, f)$ be a control system. For any $\tau, \mu \in \mathbb{R}^+$ and any input $U \ni v : [0, \tau] \to \mathbb{U}$ there exists a constant input $v_{\text{const}} : [0, \tau] \to [U]_\mu$ such that:

$$
\|v - v_{\text{const}}\|_{\infty} \leq \frac{\mu + K\tau}{2},
$$

(3.15)

where $K$ is the Lipschitz constant introduced in Definition 3.2.4.

**Proof.** We first approximate the input $v$ by the constant input $\hat{v} : [0, \tau] \to \mathbb{U}$ where $\hat{v}(t) = \frac{v(0) + v(\tau)}{2}$ for all $t \in [0, \tau]$. We then approximate $\hat{v}$ by another constant input $v_{\text{const}} : [0, \tau] \to [U]_\mu$ so that $\|\hat{v} - v_{\text{const}}\| \leq \frac{\mu}{2}$. Note that $v_{\text{const}}$ exists since $\bigcup_{q \in [U]_\mu} B_{\frac{\mu}{2}}(q)$ is a covering of $U$. Since $\hat{v}$ and $v_{\text{const}}$ are constant functions, $\|\hat{v} - v_{\text{const}}\|_{\infty} = \|\hat{v} - v_{\text{const}}\|$. Using the Lipschitz assumption for $v$, the resulting approximation error is given by:

$$
\|v - v_{\text{const}}\|_{\infty} = \|v - \hat{v} + \hat{v} - v_{\text{const}}\|_{\infty}
\leq \|v - \hat{v}\|_{\infty} + \|\hat{v} - v_{\text{const}}\|_{\infty}
\leq \|v - \hat{v}\|_{\infty} + \|\hat{v} - v_{\text{const}}\|
\leq \frac{K\tau}{2} + \frac{\mu}{2}.
$$

Proof of Theorem 3.3.1. We start by proving $S_\tau(\Sigma) \preceq S_q(\Sigma)$. Consider the relation $R \subseteq X_\tau \times X_q$ defined by $(x_\tau, x_q) \in R$ if and only if
\[ \| H_\tau(x_\tau) - H_q(x_q) \| = \| x_\tau - x_q \| \leq \varepsilon. \] Since \( X_\tau \subseteq \bigcup_{q \in [\mathbb{R}^n]} B_2(q) \), for every \( x_\tau \in X_\tau \) there exists \( x_q \in X_q \) such that:

\[ \| x_\tau - x_q \| \leq \varepsilon. \]  

Hence, \((x_\tau, x_q) \in R\) and condition (i) in Definition 3.2.8 is satisfied. Now consider any \((x_\tau, x_q) \in R\). Condition (ii) in Definition 3.2.8 is satisfied by the definition of \( R \). Let us now show that condition (iii) in Definition 3.2.8 holds.

Consider any \( u_\tau \in U_\tau \). Choose an input \( u_q \in U_q \) satisfying:

\[ \| u_\tau - u_q \| \leq \frac{\mu + K_\tau}{2}. \]  

(3.18)

Note that existence of such \( u_q \) is a consequence of Lemma 3.8.1. Consider the unique transition \( x_\tau \xrightarrow{u_\tau} x'_\tau = \xi_{x_\tau u_\tau}(\tau) \) in \( S_\tau(\Sigma) \). It follows from the \( \delta \)-FC assumption that the distance between \( x'_\tau \) and \( \xi_{x_q u_q}(\tau) \) is bounded as:

\[ \| x'_\tau - \xi_{x_q u_q}(\tau) \| \leq \beta(\varepsilon, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right). \]  

(3.19)

Since \( X_\tau \subseteq \bigcup_{q \in [\mathbb{R}^n]} B_2(q) \), there exists \( x'_q \in X_q \) such that:

\[ \| x'_\tau - x'_q \| \leq \eta. \]  

(3.20)

Using the inequalities \( \varepsilon \leq \theta \), (3.19), and (3.20), we obtain:

\[ \| \xi_{x_q u_q}(\tau) - x'_q \| = \| \xi_{x_q u_q}(\tau) - x'_\tau + x'_\tau - x'_q \| \leq \| \xi_{x_q u_q}(\tau) - x'_\tau \| + \| x'_\tau - x'_q \| \leq \beta(\varepsilon, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \eta \leq \beta(\theta, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \eta, \]

which implies the existence of \( x_q \xrightarrow{u_q} x'_q \) in \( S_q(\Sigma) \) by definition of \( S_q(\Sigma) \). Therefore, from inequality (3.20) and \( \frac{\eta}{2} \leq \varepsilon \), we conclude \((x'_\tau, x'_q) \in R\) and condition (iii) in Definition 3.2.8 holds.

Now we prove \( S_q(\Sigma) \preceq_{\delta_S} S_\tau(\Sigma) \). Consider the relation \( R \subseteq X_\tau \times X_q \). For every \( x_q \in X_q \), by choosing \( x_\tau = x_q \), we have \((x_\tau, x_q) \in R\) and condition (i) in
Definition 3.2.10 is satisfied. Now consider any \((x_\tau, x_q) \in R\). Condition (ii) in Definition 3.2.10 is satisfied by the definition of \(R\). Let us now show that condition (iii) in Definition 3.2.10 holds. Consider any \(u_q \in U_q\). Choose the input \(v_\tau = u_q\) and consider the unique \(x'_\tau = \xi_{x_\tau,v_\tau}(\tau) \in \text{Post}_{v_\tau}(x_\tau)\) in \(S_{\tau}(\Sigma)\). From the \(\delta\)-FC assumption, the distance between \(x'_\tau\) and \(\xi_{x_\tau u_q}(\tau)\) is bounded as:

\[
\|x'_\tau - \xi_{x_\tau u_q}(\tau)\| \leq \beta(\varepsilon, \tau). \tag{3.21}
\]

Since \(X_\tau \subseteq \bigcup_{q \in [\mathbb{R}^n], q} B_{\frac{\eta}{2}}(q)\), there exists \(x'_q \in X_q\) such that:

\[
\|x'_\tau - x'_q\| \leq \frac{\eta}{2}. \tag{3.22}
\]

Using the inequalities, \(\varepsilon \leq \theta\), (3.21), and (3.22), we obtain:

\[
\|\xi_{x_\tau u_q}(\tau) - x'_q\| = \|\xi_{x_\tau u_q}(\tau) - x'_\tau + x'_\tau - x'_q\|
\leq \|\xi_{x_\tau u_q}(\tau) - x'_\tau\| + \|x'_\tau - x'_q\| \leq \beta(\varepsilon, \tau) + \frac{\eta}{2}
\leq \beta(\theta, \tau) + \gamma \left(\frac{\mu + K \tau}{2}, \tau\right) + \frac{\eta}{2},
\]

which implies the existence of \(x_q \xrightarrow{u_q} x'_q\) in \(S_q(\Sigma)\) by definition of \(S_q(\Sigma)\). Therefore, from inequality (3.22) and \(\frac{\eta}{2} \leq \varepsilon\), we can conclude that \((x'_\tau, x'_q) \in R\) and condition (iii) in Definition 3.2.8 holds.

**Proof of Theorem 3.3.3.** We prove \(S_{\tau}(\Sigma) \preceq_{\Delta S} S_q(\Sigma)\). Consider the relation \(R \subseteq X_\tau \times X_q\) defined by \((x_\tau, x_q) \in R\) if and only if \(\|H_\tau(x_\tau) - H_q(x_q)\| = \|x_\tau - x_q\| \leq \varepsilon\). Since \(X_\tau \subseteq \bigcup_{q \in [\mathbb{R}^n], q} B_{\frac{\eta}{2}}(q)\), for every \(x_\tau \in X_\tau\) there exists \(x_q \in X_q\) such that:

\[
\|x_\tau - x_q\| \leq \frac{\eta}{2} \leq \varepsilon. \tag{3.23}
\]

Hence, \((x_\tau, x_q) \in R\) and condition (i) in Definition 3.2.10 is satisfied. Consider now any \((x_\tau, x_q) \in R\). Condition (ii) in Definition 3.2.10 is satisfied by the definition of \(R\). Let us now show that condition (iii) in Definition 3.2.10 holds.
Consider any $v_\tau \in U_\tau$, and choose an input $u_q \in U_q$ satisfying:
\[
\|v_\tau - u_q\|_\infty \leq \frac{\mu + K_\tau}{2}.
\tag{3.24}
\]
Note that existence of such $u_q$ is a consequence of Lemma 3.8.1. Consider the unique $x'_\tau = \xi_{x_\tau v_\tau}(\tau) \in \text{Post}_{v_\tau}(x_\tau)$ in $S_\tau(\Sigma)$. It follows from the $\delta$-FC assumption that the distance between $x'_\tau$ and $\xi_{x_q u_q}(\tau)$ is bounded as:
\[
\|x'_\tau - \xi_{x_q u_q}(\tau)\| \leq \beta(\varepsilon, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right). \tag{3.25}
\]
For all $x'_q \in \text{Post}_{u_q}(x_q)$, and based on the definition of the symbolic model, we have:
\[
\|\xi_{x_q u_q}(\tau) - x'_q\| \leq \beta(\theta, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \eta \frac{2}{2}. \tag{3.26}
\]
Using the inequalities (3.8), (3.25), and (3.26), we obtain:
\[
\|x'_\tau - x'_q\| = \|x'_\tau - \xi_{x_q u_q}(\tau) + \xi_{x_q u_q}(\tau) - x'_q\| \leq \|x'_\tau - \xi_{x_q u_q}(\tau)\| + \|\xi_{x_q u_q}(\tau) - x'_q\| \\
\leq \beta(\varepsilon, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \beta(\theta, \tau) + \gamma \left( \frac{\mu + K_\tau}{2}, \tau \right) + \eta \frac{2}{2} \leq \varepsilon.
\]
Hence $(x'_\tau, x'_q) \in R$ and condition (iii) in Definition 3.2.10 holds and we have $S_\tau(\Sigma) \lesssim_{AS} S_q(\Sigma)$.

The proof of the other direction: $S_q(\Sigma) \lesssim_{AS} S_\tau(\Sigma)$, follows from Theorem 3.3.1.

Proof of Lemma 3.4.4. We prove the result by parts. In the case when $\tilde{J}(S^*_ca, F^*_a, S_a, W_a, x_{a0}) = \infty$, the result is trivially true. Thus, we analyze the case when $\tilde{J}(S^*_ca, F^*_a, S_a, W_a, x_{a0}) < \infty$. In this case, we show that there exists a controller $S_c$ for $S_b$ such that:
\[
\tilde{J}(S_c, G, S_b, W_b, x_{b0}) \leq \tilde{J}(S^*_ca, F^*_a, S_a, W_a, x_{a0}). \tag{3.27}
\]
This is proved by showing that for every maximal behavior $y^b \in B_{(x_{c0}, x_{b0})}(S_c \times \tilde{G} S_b) \cup B_{(x_{c0}, x_{b0})}(S_c \times \tilde{G} S_b)$ there exists a maximal behavior
$y^a \in \mathcal{B}_{(x_{ca}, x_{a0})}(S_{ca}^* \times_{\mathcal{F}_a^*} S_a) \cup \mathcal{B}_{(x_{ca}, x_{a0})}(S_{ca}^* \times_{\mathcal{F}_a} S_a)$ $\varepsilon$-related to $y^b$. The proof is finalized by noting that to be optimal, the controller $(S_{cb}^*, \mathcal{F}_b^*)$ has to satisfy:

$$\bar{J}(S_{cb}^*, \mathcal{F}_b^*, S_b, W_b, x_{b0}) \leq \bar{J}(S_c^*, \mathcal{G}, S_b, W_b, x_{b0}) \leq \bar{J}(S_{ca}^*, \mathcal{F}_a^*, S_a, W_a, x_{a0})$$

for all $x_{a0} \in X_{a0}$ and $x_{b0} \in X_{b0}$ such that $(x_{a0}, x_{b0}) \in R_\varepsilon$, hence proving the result.

We start defining the controller $S_c$ for system $S_b$. Let $R_a$ be the alternating simulation relation defining the interconnection relation $\mathcal{F}_a^* = R_a^e$. We define an interconnection relation $\mathcal{G} = R_G^e$ that allows us to use the system $S_c = S_{ca}^* \times_{\mathcal{F}_a^*} S_a$ as a controller for system $S_b$. The interconnection relation $\mathcal{G} = R_G^e$ is determined by the relation:

$$R_G = \{((x_{ca}, x_a), x_b) \in (X_{ca}^* \times X_a) \times X_b \mid (x_{ca}, x_a) \in R_a \land (x_a, x_b) \in R_\varepsilon\}.$$ 

Furthermore, one can easily prove (for a detailed explanation see Proposition 11.8 in [Tabo9]) that

$$S_c \times_{\mathcal{G}} S_b \subseteq S_{ca}^* \times_{\mathcal{F}_a^*} S_a; \quad (3.28)$$

with the relation $R_{cb} \subseteq X^c \times X_c$:

$$R_{cb} = \{((x_c, x_b), x'_c) \in X^c \times X_{a0} \mid x_c = x'_c\}.$$ 

In order to show that for every maximal behavior $y^b \in \mathcal{B}_{(x_{a0}, x_{b0})}(S_c \times_{\mathcal{G}}^e S_b) \cup \mathcal{B}_{(x_{a0}, x_{b0})}(S_c \times_{\mathcal{G}} S_b)$ there exists an $\varepsilon$-related maximal behavior $y^a \in \mathcal{B}_{(x_{ca}, x_{a0})}(S_{ca}^* \times_{\mathcal{F}_a^*} S_a) \cup \mathcal{B}_{(x_{ca}, x_{a0})}(S_{ca}^* \times_{\mathcal{F}_a} S_a)$, we first make the following remark: for any pair $(x_a, x_b) \in R_\varepsilon$, by the definition of alternating simulation relation, if $U_a(x_a) \neq \emptyset$ then $U_b(x_b) \neq \emptyset$. From the definition of $\mathcal{G}$ it follows that for all $((x_{ca}, x_a), x_b) \in X_{ca}$ the pair $(x_a, x_b)$ belongs to $R_\varepsilon$. Thus, for any pair of related states $(x_a, x_b) \in R_\varepsilon$ there exists $x_G \in X_G$, namely $(x_c, x_b)$, with $x_c = (x_{ca}, x_a)$, so that $U_c(x_c) \neq \emptyset \implies U_G(x_G) \neq \emptyset$. The existence of the simulation relation (3.28) implies that for every behavior $y^b$ there exists an $\varepsilon$-related
behavior \( y^a \). Any infinite behavior is a maximal behavior, and thus we already know that for every (maximal) infinite behavior \( y^b \) there exists an \( \varepsilon \)-related (maximal) infinite behavior \( y^a \). Moreover, if \( y^b \) is a maximal finite behavior of length \( l \), the set of inputs \( U_G(y^b) \) is empty. As shown before, this implies that \( U_c(y^a) = \emptyset \), and thus \( y^a \) is also maximal, where \( y^a \) is the corresponding behavior of \( S^*_c \times F^*_a S_a \) \( \varepsilon \)-related to \( y^b \).

We now show that (3.27) holds. For any initial state \( x_{a0} \) there exists an initial controller state \( x_{ca0} \in R^{-1}_a(x_{a0}) \) of \( S^*_c \), such that every maximal behavior \( y^a \in B(x_{ca0},x_{a0})(S^*_c \times F^*_a S_a) \cup B^G(x_{ca0},x_{a0})(S^*_c \times F^*_a S_a) \) reaches a state \( x_a \in W_a \) in the worst case after \( \tilde{J}(S^*_c, F^*_a, S_a, W_a, x_{a0}) \) steps. We assume in what follows that the controller is initialized at that \( x_{ca0} \). Thus, as maximal behaviors of \( S^*_c \times \varepsilon G S_b \) are related by \( R_{cb} \) to maximal behaviors of \( S^*_c \times F^*_a S_a \), for every maximal behavior \( y^b \in B(x_{a0},x_{b0})(S^*_c \times F^*_a S_a) \cup B^G(x_{a0},x_{b0})(S^*_c \times F^*_a S_a) \) reaches some state \( x_b \in R_{\varepsilon}(W_a) \) in at most \( \tilde{J}(S^*_c, F^*_a, S_a, W_a, x_{a0}) \) steps. But then, from the second assumption, \( x_b \in R_{\varepsilon}(W_a) \) implies that \( x_b \in W_b \) and we have that

\[
\tilde{J}(S^*_c, G, S_b, W_b, x_{b0}) \leq \tilde{J}(S^*_c, F^*_a, S_a, W_a, x_{a0})
\]

for all \( x_{a0} \in X_{a0} \) and \( x_{b0} \in X_{b0} \) such that \( (x_{a0}, x_{b0}) \in R_{\varepsilon} \).

\( \square \)

**Proof of Theorem 3.4.6.** Note that \( S_b \preceq_{AS} S_{d(a)} \), by the assumed relation and both systems being deterministic. Also note that, by definition, \( R([W]_R) \subseteq W \) and \( R^{-1}(W) \subseteq [W]_R \). Then the proof follows from Lemma 3.4.4. \( \square \)
CHAPTER 4

Conclusion

In this thesis I have studied some design problems for Networked Cyber-Physical Systems from two different perspectives. First, in Chapter 2, I addressed the implementation of controllers over communication networks. I provided two different aperiodic implementations that reduce the communication burden of networked control systems. In particular, I focused on WSAN, where communication is specially costly, although the same techniques and results can be beneficial in other kinds of networked systems. On Chapter 3, I addressed the design of controllers for complex specifications requiring guarantees of correct operation. I introduced model abstractions for control systems that reduce them to finite state machines. These abstractions, with a countable number of states, ease the design of controllers for complex specifications by enabling the use algorithmic solutions developed for discrete event systems and games on automata. Furthermore, I studied how such abstractions can also be employed to solve time-optimal control problems that deliver guaranteed performance bounds. This theoretical work has also been complemented with the development of a Matlab toolbox, named Pessoa, implementing both the construction of symbolic abstractions for control systems and the synthesis of correct-by-design controllers for several specifications given as a small subset of Linear Temporal Logics formulae.

The contributions made in this thesis represent only steps towards more ambitious goals. The implementations of networked control systems can only benefit
from the contributions described in this thesis if there are communication protocols that can support them, and scheduling algorithms capable of accommodating aperiodic control tasks. Early studies of these issues can already be found [TFJ10], [AT09], but much remains to be done to transfer these techniques to real deployments. Similarly, while many problems can already be solved relying on the techniques presented in Chapter 3, further research is required to enable their adoption in industry. The currently available symbolic abstractions and correct-by-design synthesis algorithms only apply to small dimensional systems or to constricted classes of dynamics. The results presented in this thesis partly solves the latter limitation, while still suffering from the curse of dimensionality, which limits their application to relatively small dimensional systems. Yet one more challenge for the future is to solve this issue, either by exploiting the given specifications to construct smaller symbolic abstractions specifically tailored to the problem addressed, or by developing compositional methods [KS10] capable of applying the maxim of divide and conquer to the synthesis of controllers.

Future work should also include extensions to the results developed for time-optimal control employing symbolic abstractions. Solutions to more quantitative specifications including general optimal control problems and robust controller designs are such desirable extensions. Finally, while the two Chapters of this thesis address problems encountered in the design and implementation of Networked Cyber-Physical Systems, and their complementing nature seems obvious, how to combine these techniques is left for future research.
References


http://trac.parades.rm.cnr.it/ariadne/.


[Hyba] “HybridSal.”
http://sal.csl.sri.com/hybridsal/.

[Hybb] “Hybrid Toolbox.”


[LTL] “LTLCon.”
http://iasi.bu.edu/~software/LTL-control.htm


http://www.cyphylab.ee.ucla.edu/pessoa.


[Zig] “ZigBee-Alliance.”

