UNIVERSITÀ DEGLI STUDI DELL'AQUILA
DEPARTMENT OF INFORMATION ENGINEERING,
COMPUTER SCIENCE AND MATHEMATICS

PhD Program in **Information and Communication Technology**

Curriculum **Systems Engineering telecommunications and HW/SW platforms**

SSD **ING/INF-04**

XXXIV cycle

Control of Cyber-Physical Systems: a Formal Method approach

Tommaso Masciulli

Advisor Prof. **Giordano Pola**

Coordinator Prof. **Vittorio Cortellessa**

L'Aquila 2022

I dedicate this to my parents

Experience is the hardest kind of teacher;

it gives you the test first and the lesson afterward.

— Oscar Wilde

Abstract

Over the past few decades, the rapid evolution of computing has brought to the born of more complex and detailed paradigms of system, named Cyber-Physical Systems (CPS), i.e. systems where physical processes, generally described by continuous dynamics, interact with computing units, generally described by (discrete) models of computation, using a (nonideal) communication infrastructure. The generality of CPS to achieve different and complex scenarios translates in a difficulty to provide systematic methods to their analysis and control. A promising solution is given by hybrid systems, that are dynamical models that combine behaviours of purely continuous dynamics with discrete dynamics. The rich and complex behaviour of such models poses difficulties about how to approach the design of control systems.

In the last twenty years, researchers working in the area of computer science and control theory have explored formal methods as an automatic tool for addressing analysis and control design of this complex kind of systems. Central to this approach is the construction of *symbolic system* that approximate purely continuous or hybrid plants and preserve the same properties of the Cyber-Physical Systems while hiding the details that are of no interest. A symbolic system is an abstract description of a purely continuous or hybrid system where each state corresponds to an aggregate of continuous/hybrid states and each label to an aggregate of continuous/hybrid inputs. The relevance of this approach is corroborated by considering complex logic specifications and mathematical models that can directly incorporate constraints on hardware and software architectures. The contribution of this thesis is to enhance the amount of possible scenario accounted by symbolic systems.

Table of Contents

Introduction

Technology advances have made almost any system of interest smart, i.e. capable of carrying out advanced task in a semi-independent or independent fashion. The mathematical arsenal on differential and difference equations that has been employed to model and study the processes governed by the laws of nature results inadequate or simply inappropriate for several scenarios. Below some examples of possible scenarios:

(Automotive) Vehicles now include hierarchically distributed computing systems which coordinate several distinct control functions, like transmission controls, cruise control, antilock brakes, etc. Furthermore, with the recent computing power the first self-driving vehicles begin to be created.

(Air-traffic control) Any airplane requires several controllers to correctly accomplish its tasks, but one of the most difficult challenges is to control the air-traffic system. This environment not only considers the interaction of complex systems with complex specifications but also the human presence.

(Manufacturing plants and power plants) There are several scenarios too dangerous or too hard for the human presence, so robotics offers a valid solution to this problem. This improvement requires more complex systems to model and more complex specifications to achieve.

The integration of physical processes with networked computing has led to a new generation of engineered systems, named Cyber-Physical Systems (CPSs), i.e. systems where physical processes, generally described by continuous dynamics, interact with computing units, generally described by (discrete) models of computation, using a (nonideal) communication infrastructure.

CPSs are large-scale, complex, heterogeneous, distributed and networked systems where physical processes interact with distributed computing units through communication networks. Future cyber-physical systems are expected to overcome those of today in terms of adaptability, autonomy, efficiency, functionality, reliability, safety and usability. Research advances in cyber-physical systems are expected to transform our world with systems that are more precise (e.g. robotic surgery), respond more quickly, augment human capabilities, work in dangerous or inaccessible environments (e.g. autonomous systems for search and rescue, firefighting and exploration), provide large-scale, distributed coordination (e.g. automated traffic control), are highly efficient (e.g. zero-net energy buildings), enhance wellbeing (e.g. assistive technologies and ubiquitous healthcare monitoring). The generality of CPS to achieve different and complex scenarios translates in a difficulty to provide systematic methods to their analysis and control. Some critical aspects are:

Heterogeneity: CPSs are heterogeneous, given the different mathematical domains we need to describe them.

Complexity: CPSs may consist of many components, each of which can hierarchically contain a large set of subcomponents.

Specifications: specifications for CPSs are themselves heterogeneous involving functional, architectural and implementation directives.

0.1 Hybrid systems

A comprehensive and general formal framework is difficult to find. A good solution is given by the *Hybrid Systems*: models that consider a huge number of hierarchical and complex process. Hybrid systems are dynamical models that

combine behaviours of purely continuous dynamics with discrete dynamics.
More formally, a hybrid system consists of an automaton with a finite number of discrete states (modes), and continuous dynamics associated to each mode. The model of the automaton is formalized through a Discrete-Event System (DES), while continuous dynamics are modelled by differential equations. The state of such a system is composed of a discrete component and a continuous component, and the evolution is divided between continuous and discrete evolution, where one influences the other. The continuous one depends on the differential equations of the current discrete state so the transition from a discrete state to another change the set of differential equations. Likewise, the discrete evolution depends on the Domain of the current discrete state and the continuous state. The mutual interaction of the continuous part with the discrete part of a hybrid system expresses the power of this technique. Examples of such complex systems are widely employed in engineering, such as heating and cooling system, continuous systems with phased operations or controlled by a discrete logic, switched electric circuits, embedded systems, etc. The rich and complex behaviour of such models poses difficulties about how to approach their analysis and the design of control systems.

0.2 Discrete abstraction

In the last twenty years, researchers working in the area of computer science and control theory has explored formal methods as an automatic tool for addressing analysis and control design of hybrid systems, otherwise too complicate to manage in any other way. Central to this approach is the construction of *Symbolic Systems* that approximate purely continuous or hybrid plants and preserve the properties being analyzed while hiding the details that are of no interest. A symbolic

system is an abstract description of a purely continuous or hybrid system where each state corresponds to an aggregate of continuous/hybrid states and each label to an aggregate of continuous/hybrid inputs. The relevance of this approach is also corroborated by the following advantages:

Symbolic models belong to the same mathematical class as the models used to describe software and hardware in synthesis and verification methods developed in the computer science and engineering community. Hence, they can be used to translate and adapt these methods to hybrid systems thus, allowing one to take into consideration constraints on hardware and software architectures in a natural way.

Complex logic specifications, such as reachability with obstacle avoidance, motion planning, periodic orbits, state-based switching, sequences of smaller tasks that need to be performed according to a given order, are difficult for traditional control synthesis methods, while using symbolic models we can leverage correct-by-construction techniques developed in the computer science and engineering community.

0.3 Thesis outline

This section details the organization of the thesis and the contribution to this research area. The remainder of the text is organised as follows:

Chapter 1 concerns the definitions of deterministic and nondeterministic system used in the following and the fundamental concepts of trajectories associated with it.

Chapter 2 recalls the concepts of languages and the meaningful operators on them.

Chapter 3 considers unary and binary operations that alter the state transition diagram of a system and some bisimulation equivalences.
Below the mainly references of these first three chapters:

- C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Boston, MA, USA: Kluwer Acad. Publ., 1999.
- P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach.* Springer Science and Business Media, 2009.
- G. Pola, and M. D. Di Benedetto, *Control of Cyber-Physical-Systems with logic specifications: A formal methods approach.* Annual Reviews in Control, vol. 47: 178-192, 2019.

Chapter 4 considers a symbolic control problem where the system does not know all the specification a priori, but has only the knowledge of a first part of it; then, at some time the system will obtain a second specification for which the controller needs to be reconfigured. The example at the end of the chapter shows the advantages of the proposed algorithms. The results of this chapter are based on the works:

- T. Masciulli, and G. Pola, *Symbolic control design of incrementally stable nonlinear systems with dynamic regular language specifications.* Automatica, vol. 130, 2021.
- T. Masciulli, and G. Pola, *On symbolic control design of nonlinear systems with dynamic regular language specifications.*
  IFAC-PapersOnLine, vol. 53(2): 1844-1849, 2020.

Chapter 5 addresses control design of nondeterministic finite state systems with reachability specifications and reach-avoid specifications with also the identification of the state that has been reached in the target set. The strength

of this approach lies in the generality of the systems considered, nondeterministic systems with a non-injective output function, and in the final theorem, an if and only if condition that ensures the unicity of this approach. This chapter is mainly based on the works:

- T. Masciulli, G. Pola, E. De Santis, and M. D. Di Benedetto, *Output Feedback Reachability of Controlled–Observable States for Nondeterministic Finite–State Systems.* IEEE Control Systems Letters, vol. 6: 464-469, 2022.

- D. A. Ajeleye, T. Masciulli, and G. Pola, *Output Feedback Control of Nondeterministic Finite–State Systems with Reach–Avoid Specifications.* 29th Mediterranean Conference on Control and Automation, 2022.

Chapter 6 addresses data-driven control design of an unknown plant, apart from a finite set of experiments. The specifications are assumed to be given in terms of desired regular language, and the proposed controller enforces the specifications on the plant, up to an error that can be chosen as small as desired. The application at the end of the chapter to the artificial pancreas highlights the potential of the proposed technique. This chapter collects results from the following papers:

- G. Pola, T. Masciulli, E. De Santis, and M. D. Di Benedetto, *Data-driven controller synthesis for abstract systems with regular language specifications.* Automatica, vol. 134, 2021.

- G. Pola, T. Masciulli, E. De Santis, and M. D. Di Benedetto, *On data-driven controller synthesis with regular language specifications.* IFAC-PapersOnLine, vol. 53(2): 3928-3933, 2020.

Chapter 7 gives some concluding remarks and outlook for future works.

Notations

Some recall about set theory:

- symbol $\mathbb{N}$ denotes the set of positive integers;
- symbol $\mathbb{N}_0$ denotes the set of nonnegative integers;
- symbol $\mathbb{Z}$ denotes the set of integers;
- symbol $\mathbb{R}$ denotes the set of real numbers;
- symbol $\mathbb{R}^+$ denotes the set of positive real numbers;
- symbol $\mathbb{R}_0^+$ denotes the set of nonnegative real numbers.

Given $a, b \in \mathbb{R}$, we denote by

- $[a, b]$ the closed set of real numbers between a and b;
- (a, b) the open set of real numbers between a and b.

Given $a, b \in \mathbb{Z}$, we denote $[a; b] = [a, b] \cap \mathbb{Z}$.

Given $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the floor of real x, i.e. such that $\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \leq x\}$.

Given $x \in \mathbb{R}^n$, we denote by

- $\|x\|$ the infinity norm of x;
- $\|x\|_2$ the euclidean norm of x.

By following standard set theory, we consider sets with no occurrence of same elements, e.g. we consider set $\{1, 2\}$ instead of set $\{1, 1, 2\}$. Moreover:

- Symbol $card(X)$ denotes the cardinality of a finite set X;

- symbol 2^X denotes the set of subsets of a set X, i.e. the power set of X.

Given $a \in \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, the symbol aX denotes the set

$$\{y \in \mathbb{R}^n \mid \exists x \in X \text{ s.t. } y = ax\}.$$

Given a pair of sets X and Y, let $X \setminus Y = \{x \in X \mid x \notin Y\}$. Given a set $Y \subseteq X$, symbol Y^c denotes the complement of Y in X, i.e. $Y^c = X \setminus Y$.

Given a pair of sets X and Y and a relation $\mathcal{R} \subseteq X \times Y$, the symbol $\mathcal{R}^{-1}$ denotes the inverse relation of $\mathcal{R}$, i.e. $\mathcal{R}^{-1} = \{(y, x) \in Y \times X \mid (x, y) \in \mathcal{R}\}$. Given $X' \subseteq X$ and $Y' \subseteq Y$, we denote $\mathcal{R}(X') = \{y \in Y \mid \exists x \in X' \text{ s.t. } (x, y) \in \mathcal{R}\}$ and $\mathcal{R}^{-1}(Y') = \{x \in X \mid \exists y \in Y' \text{ s.t. } (x, y) \in \mathcal{R}\}$.

Given a function $f : X \to Y$, the symbol $f^{-1} : Y \to 2^X$ denotes the inverse map of f, i.e. $f^{-1}(y) = \{x \in X \mid y = f(x)\}$ for all y in the co-domain of f. Given $f : X \to Y$ and $X' \subseteq X$ the symbol $f(X')$ denotes the image of X' through f, i.e. $f(X') = \{y \in Y \mid \exists x \in X' \text{ s.t. } y = f(x)\}$. Similarly, given $Y' \subseteq Y$ the symbol $f^{-1}(Y')$ denotes the inverse image of Y' through f, i.e. $f^{-1}(Y') = \{x \in X \mid f(x) \in Y'\}$. Given $X' \subset X$ the symbol $f|_{X'}$ denotes the restriction of f to X' that is $f|_{X'} : X' \to Y$ such that $f|_{X'}(x') = f(x')$ for all $x' \in X'$. Given two functions $f : X \to Y$ and $g : Y \to Z$ we denote by $f \circ g : X \to Z$ the composition function of f and g, defined by $(f \circ g)(x) = f(g(x))$, for all $x \in X$.

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 $\gamma(0) = 0$; function γ is said to belong to class $\mathcal{K}_\infty$ if $\gamma \in \mathcal{K}$ and $\gamma(r) \to \infty$ as $r \to \infty$. A continuous function $\beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is said to belong to class $\mathcal{KL}$ if, for each fixed s, the map $\beta(r, s)$ belongs to class $\mathcal{K}_\infty$ with respect to r and, for each fixed r, the function $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to \infty$.

Given a string x we denote by $x(i)$ the i–th element of x and by $x(end)$ the last element of x.

Given $\theta \in \mathbb{R}^+$ and $x \in \mathbb{R}^n$, we denote by

$$\mathcal{B}^n_{[-\theta,\theta[}(x) = \{y \in \mathbb{R}^n \mid y(i) \in [-\theta + x(i), \theta + x(i)[,\ i \in [1; n]\},$$

corresponding to the left–closed right–open hyper–cube in $\mathbb{R}^n$ centered at x and with side $2\,\theta$.

Given $n \in \mathbb{N}_0$ and quantization parameter $\theta \in \mathbb{R}^+$, the quantizer in $\mathbb{R}^n$ with accuracy θ is a function $[\,\cdot\,]^n_\theta : \mathbb{R}^n \to 2\theta\mathbb{Z}^n$, associating to any $x \in \mathbb{R}^n$ the unique vector $[x]^n_\theta \in 2\theta\mathbb{Z}^n$ such that $x \in \mathcal{B}^n_{[-\theta,\theta[}([x]^n_\theta)$. Definition above naturally extends to sets $X \subseteq \mathbb{R}^n$ when $[X]^n_\theta$ is interpreted as the image of X through function $[\,\cdot\,]^n_\theta$.

1 | Control Systems

In this chapter the definitions of system used throughout the text and the fundamental concepts associated with it are detailed. Moreover, a useful classification of systems is provided.

1.1 The concept of system

System is a fundamental concept for several scientific area, widely used and hence with several definitions. We can provide three representative definitions found in the literature:

- An aggregation or assemblage of things so combined by nature or man as to form an integral or complex whole (Encyclopedia Americana).
- A regularly interacting or interdependent group of items forming a unified whole (Webster's Dictionary).
- A combination of components that act together to perform a function not possible with any of the individual parts (IEEE Standard Dictionary of Electrical and Electronic Terms).

Since our main goal is to design controllers to accomplish some dedicated tasks, the purely qualitative definitions given above are inadequate. Hence, we need to develop some mathematical means for describing the behaviour of the system itself. In fact, we are seeking for the mathematical object called *model*. Strictly speaking, system denotes "something real", whereas model is an "abstraction" (a set of mathematical equations). Often, the model only approximates the true behaviour of the system. However, once we are convinced we have obtained a good model, this distinction is usually dropped, and the terms system and model

are used interchangeably.

At this point, including the classical notation, we introduce our notation. Since model and system are often interchangeably we denote:

- plant as "something real" and
- system as "set of equations".

Let us move on detail the shape and boundary of this set of equations.

Definition 1 *A system S is a tuple $(X, x_0, U, f, X_m, \Gamma, Y, H)$, where*

- *X is the set of states;*
- *$x_0 \in X$ is the initial state;*
- *U is the set of inputs;*
- *$f : X \times U \to X$ is the transition function, $f(x, u) = x'$ means that from state x with the input u the system S reaches the state x';*
- *$X_m \subseteq X$ is the set of marked states;*
- *$\Gamma : X \to 2^U$ is the active inputs function defined by $\Gamma(x) = \{u \in U \mid f(x, u)$ is defined$\}$;*
- *Y is the set of outputs;*
- *$H : X \to Y$ is the output function.*

In order to better understand definition above we provide the following

Example 1 *Consider system S as in Definition 1, where*

- *$X = \{x_0, x_1, x_2, x_3\}$, the states;*
- *x_0, the initial state;*

- $U = \{u, v\}$, *the inputs;*
- *the transition function* $f : X \times U \to X$ *is detailed as a list of transitions:* $x_1 = f(x_0, u)$, $x_3 = f(x_0, v)$, $x_1 = f(x_2, u) = f(x_2, v)$ *and* $x_3 = f(x_3, u)$*;*
- $X_m = \{x_1\}$, *the marked state;*
- *the active inputs function* $\Gamma : X \to 2^U$ *is detailed as the transition function:* $\Gamma(x_0) = \{u, v\}$, $\Gamma(x_1) = \varnothing$, $\Gamma(x_2) = \{u, v\}$ *and* $\Gamma(x_3) = \{u\}$*;*
- $Y = \{a, b, c\}$, *the outputs;*
- *the output function* $H : X \to Y$ *is detailed as the transition function:* $H(x_0) = a$, $H(x_1) = a$, $H(x_2) = b$ *and* $H(x_3) = c$.

All these information can be encoded in a graph, as in Figure 1.1, where

- *circles are the states;*
- *the state with the void arrow is the initial state;*
- *arrows are the transitions and the labels over them are the inputs;*
- *the states with the double circles are the marked states;*
- *labels over the states are the outputs.*

Three observations rose regarding Example 1. First, the value of a state of the system may not change after an enabled input in that state, also known as "self loop". Second, the system may evolve in the exact same way after two distinct inputs. Third, the function f may be a partial function on its domain $X \times U$, that is, not all inputs in U cause a transition for every state of X.

At this stage we did not provide any assumptions over the set of states, the set of inputs and the set of outputs. Indeed, a system S as in Definition 1 is classified accordingly with the following list. A system S is said to be

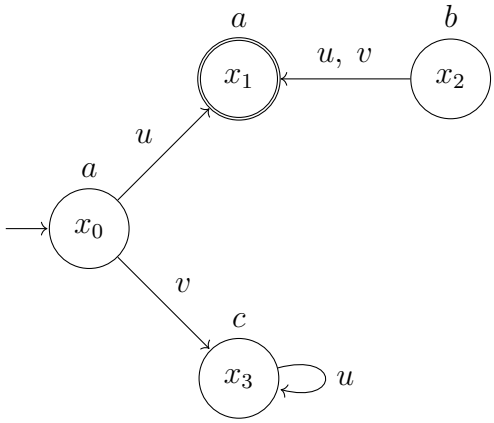

Figure 1.1: Graph representation of system S of Example 1.

- *symbolic* if the set of states and the set of inputs are finite;
- *metric* if the set of outputs is equipped with a metric $d : Y \times Y \to \mathbb{R}_0^+$;
- *nonblocking* if from each state it is possible to reach a marked state.

As shown in Example 1 it is immediately seen that in some applications the transition function f of a symbolic system is a discrete function with a finite number of transitions. Hence, we can extend the definition of cardinality to transition function.

Definition 2 *Given a symbolic system S as in Definition 1 we say that*

$$card(f) := \sum_{x \in X} card(\Gamma(x)).$$

if X and U are finite sets.

For later purposes, given a system S as in Definition 1 we define the notions of trajectories.

Definition 3 *An input run $\{u_i\}_{i=1,\dots,n}$ is a finite sequence of elements in U with $n \in \mathbb{N}_0$.*

Sometimes, we will represent a sequence $\{a_i\}_{i=1,\dots,n}$ over a finite set A for some $n \in \mathbb{N}_0$ as a string $a_1\, a_2\, \dots\, a_n$. We point out that if $n = 0$ the sequence is empty, usually represented with the symbol ε.

For sake of convenience and for future purposes, we always extend the domain of the transition function f from $X \times U$ to $X \times U^*$ in the following recursive manner:

$$\begin{aligned} f(x, \varepsilon) &:= x \\ f(x, s\, u) &:= f(f(x, s), u) \end{aligned}$$

where s is a string in U^*, the set of strings over U, and $u \in \Gamma(x) \subseteq U$.

A careful reader would notice that the extension above of the function f raises a problem, it requires a sequence of enabled inputs. Let's see an examples over graph in Fig. 1.1. Consider the input run $v\, v$, then $f(x_0, v\, v)$ is not defined in fact $f(x_0, v\, v) = f(f(x_0, v), v)$ where $f(x_0, v) = x_3$ but $f(x_3, v)$ is not defined. With the following definition we avoid this problem considering a more general state evolution, moreover we keep track of the states considered in the evolution.

Definition 4 *A state run (or evolution) $\{x_i\}_{i=0,\dots,n}$ is a finite sequence of elements in X with $x_0 \in X_0$ such that for some input runs $\{u_k\}_{k=1,\dots,p}$ with an index sequence $0 = k_0 < k_1 < \cdots < k_n \leq p$ we get*

- *$u_k \notin \Gamma(x_i)$ for all $k \in (k_i; k_{i+1})$ and $i = 0, \dots, n-1$, and $u_k \notin \Gamma(x_n)$ for all $k > k_n$;*
- *$x_{i+1} = f(x_i, u_{k_{i+1}})$ for all $i = 0, \dots, n-1$.*

We say that a state run is generated by (associated to) an input run if Definition 4 holds. We say that a state run is strictly generated by (strictly associated to) an input run if Definition 4 holds with $p = n$.

Example 2 *Consider system S as in Definition 1 detailed as in Figure 1.2.*

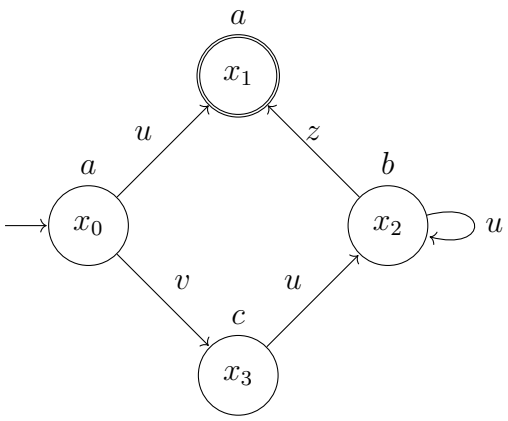

Figure 1.2: System S of Example 2.

Consider for example the input run $z\,z\,v\,z\,v\,u\,u\,z\,v$; the corresponding state run is $x_0\,x_3\,x_2\,x_2\,x_1$. In fact, we can separate the input run as in this way $z\,z$, v, $z\,v$, u, u, z, v, where: the first sequence $z\,z$ does not affect state x_0; after the first v the system evolves from x_0 to x_3, i.e. $x_3 = f(x_0, v)$; the sequence $z\,v$ does not affect state x_3, hence nothing happens; the two inputs u cause the evolution from x_3 to x_2 and then from x_2 to x_2; the input z causes the evolution from x_2 to x_1; now, since nothing can happen from x_1, the last v causes no evolution.

Definition 5 *An output run $\{y_i\}_{i=0,\dots,n}$ is a finite sequence of elements in Y such that $y_i = H(x_i)$ for all $i = 0, \dots, n$ and some state runs $\{x_i\}_{i=0,\dots,n}$.*

In the following we will also use the notation of languages of Section 2.1 to denote the three trajectories

- input run: $u_1\,u_2\,\dots\,u_p$;
- state run: $x_0\,x_1\,\dots\,x_n$;
- output run: $y_0\,y_1\,\dots\,y_n$.

We point out that the inclusion of Γ in the definition of S is superfluous in the sense that Γ is derived from f. For this reason, we will sometimes omit explicitly writing Γ when specifying a system where the active input function is not instrumental to the discussion. One of the reasons why we use $\Gamma(x)$ is to help distinguish between inputs u that are feasible at x but cause no state transition, and inputs u that are not feasible at x, that is, $f(x, u)$ is not defined.

1.2 Nondeterminism

The Definition 1 of Section 1.1 takes into account only deterministic systems, for future purposes we need to consider a more general definition of system. It is first necessary to understand the definition of determinism, let us introduce the following

Definition 6 *A system S is deterministic if for any state x and input run there exists one and only one state run starting from x.*

In other words, it is always possible to infer the state run from the only knowledge of the input run. From this we deduce that a system S of Definition 1 is said to be deterministic because f is a function (not a map) from $X \times U$ to X, namely, for any state x there exists at most one successor of x for any input in U. In contrast, the transition structure of a nondeterministic system is defined by means of a function from $X \times U \to 2^X$; in this case, there can be multiple transitions starting from a state with the same input.

Moreover, there are two other ways to generalize a deterministic system in Definition 1 into a nondeterministic one:

- we can consider several initial states, i.e. $X_0 \subseteq X$;

- we can have the empty input (denoted with ε, this symbol is also used to denote the empty output) and a state x for which $f(x, \varepsilon) \neq x$.

We point out that a system is nondeterministic if at least one of the three previous conditions holds. Furthermore, the nondeterminism given by multiple initial states can be included in the case of ε transitions. Indeed, we can set a dummy state x_D as the initial state and then we set $f(x_D, \varepsilon) := X_0$ where X_0 is the set of initial states.

Despite this possibility we prefer to consider the case of several initial states and to avoid the case of empty input. There are two reasons for this: since the goal of this work is to find a controller, we need to consider controllable inputs; sometimes it is possible to choose the initial states.

For the sake of clarity and for future purposes we provide the most common definition of nondeterministic system for this thesis.

Definition 7 *A system S is a tuple $(X, X_0, U, f, X_m, \Gamma, Y, H)$, where*

- *X is the set of states;*
- *$X_0 \subseteq X$ is the set of initial states;*
- *U is the set of inputs;*
- *$f : X \times U \to 2^X$ is the transition map, $x' \in f(x, u)$ means that from state x with the input u the system S may reach the state x';*
- *$X_m \subseteq X$ is the set of marked states;*
- *$\Gamma : X \to 2^U$ is the active inputs function defined by $\Gamma(x) = \{u \in U \mid f(x, u)$ is defined$\}$;*
- *Y is the set of outputs;*

- $H : X \to Y$ *is the output function.*

Example 3 *Consider system S as in Definition 7, where*

- $X = \{x_0, x_1, x_2, x_3, x_4\}$*, the states;*
- $X_0 = \{x_0, x_1\}$*, the initial states;*
- $U = \{u, v\}$*, the inputs;*
- *the transition map* $f : X \times U \to 2^X$ *is detailed as a list of transitions:* $\{x_2, x_3\} = f(x_0, u)$*,* $\{x_3\} = f(x_1, v)$ *and* $\{x_2\} = f(x_4, u)$*;*
- $X_m = \{x_2\}$*, the marked states set;*
- *the active inputs function* $\Gamma : X \to 2^U$ *is detailed as follows:* $\Gamma(x_0) = \{u\}$*,* $\Gamma(x_1) = \{v\}$*,* $\Gamma(x_2) = \Gamma(x_3) = \varnothing$ *and* $\Gamma(x_4) = \{u\}$*;*
- $Y = \{a, b, c\}$*, the output set;*
- *the output function* $H : X \to Y$ *is detailed as follows:* $H(x_0) = a$*,* $H(x_1) = a$*,* $H(x_2) = a$*,*$H(x_3) = c$ *and* $H(x_4) = b$*.*

All this information can be encoded in a graph, as in Figure 1.3, with the same notation as in Example 1.

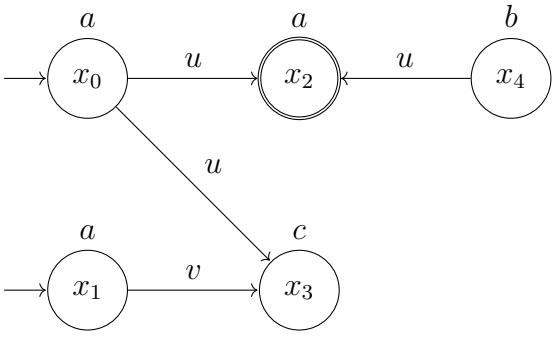

Figure 1.3: Graph representation of system S of Example 3.

For later purposes, as in the case of deterministic systems, we extend the transition function f to the extended case $f^{ext} : 2^X \times U^* \to 2^X$, as follows.

For all singletons $\{x\} \in 2^X$ and $u \in U$, we define:

$$f^{ext}(\{x\}, u) := \begin{cases} f(x,u), & \text{if } u \in \Gamma(x); \\ \{x\}, & \text{otherwise.} \end{cases}$$

For all $X' \in 2^X$ and $u \in U$, we define:

$$f^{ext}(X', u) := \bigcup_{x \in X'} f^{ext}(\{x\}, u).$$

For all $X' \in 2^X$ and $\overline{u}\, u$, with $\overline{u} \in U^*$, $u \in U$, we define:

$$f^{ext}(X', \overline{u}\, u) := f^{ext}(f^{ext}(X', \overline{u}), u).$$

Even for Definition 7 we can give concept of cardinality and trajectories with some adjustments.

Definition 8 *Given a symbolic system S as in Definition 1 we say that*

$$card(f) := \sum_{x \in X} \sum_{u \in \Gamma(x)} card(f(x,u)).$$

Remark 1 *We point out that this definition could be used for deterministic systems setting: $card(f(x,u)) = 1$ if $f(x,u)$ is defined, and $card(f(x,u)) = 0$ otherwise.*

The notions of input and output runs are the same as those given in the previous

section; instead, the notion of state run needs to be reformulated as follows.

Definition 9 *A state run (or evolution)* $\{x_i\}_{i=0,\dots,n}$ *is a finite sequence of elements in* X *with* $x_0 \in X_0$ *such that for some input runs* $\{u_k\}_{k=1,\dots,p}$ *with an index sequence* $0 = k_0 < k_1 < \cdots < k_n \leq p$ *we get*

- $u_k \notin \Gamma(x_i)$ *for all* $k \in (k_i; k_{i+1})$ *and* $i = 0, \dots, n-1$*, and* $u_k \notin \Gamma(x_n)$ *for all* $k > k_n$*;*
- $x_{i+1} \in f(x_i, u_{k_{i+1}})$ *for all* $i = 0, \dots, n-1$.

We say that a state run is generated by (associated to) an input run if Definition 9 holds.

The following notion will be useful in the sequel.

Definition 10 *A system* S *is output deterministic if for any output run there exists one and only one state run.*

From this definition we deduce that a system $S = (X, X_0, U, f, X_m, \Gamma, Y, H)$ is output deterministic if for all $x \in X$ and $u_1, u_2 \in \Gamma(x)$ we get

$$H(x_1) \neq H(x_2),\ \forall x_1, x_2 \in f(x, u_1) \cup f(x, u_2) \text{ and } x_1 \neq x_2.$$

We point out that Definition 10 can be referred either to systems of Definition 1 or 7; namely, there is no connection between nondeterminism and output nondeterminism. To better understand this concept, let's give the following example.

Example 4 *We have already seen some examples of deterministic and output deterministic systems, i.e. systems in Figure 1.1 and 1.2. Let us now consider the three different systems in Figure 1.4.*

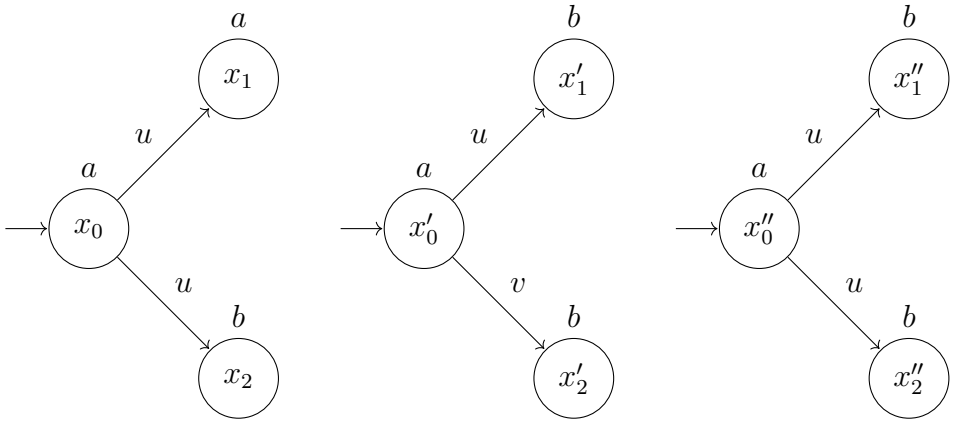

Figure 1.4: System on the left: nondeterministic but output deterministic. System in the center: deterministic but output nondeterministic system. System on the right: nondeterministic and output nondeterministic.

The system on the left is nondeterministic because $f(x_0, u) = \{x_1, x_2\}$*, namely there is at least one state for which different evolutions can occur with one input; however, the system is output deterministic, in fact, for any output run* a*,* $a\,a$ *and* $a\,b$ *we can associate one and only one state run:* x_0*,* $x_0\,x_1$ *and* $x_0\,x_2$ *respectively.*

The system in the center is deterministic but output nondeterministic, in fact, to the output run $a\,b$ *we can associate either the state run* $x'_0\,x'_1$ *or* $x'_0\,x'_2$*.*

For all of the above, the system on the right is nondeterministic and output nondeterministic.

2 | Languages

We have seen that the evolution of a system is not driven by a clock. Hence, when we consider the state evolution of a system, our first concern is with the sequence of states visited and the associated inputs causing these state transitions. In other words, we do not care of when the system enters in a particular state or how long the system remains at that state. We will characterize the dynamic of the system through sequences of inputs (outputs). A sequence of that form specifies the order in which various events occur over time, but it does not provide the time instants associated with the occurrence of these events. Consequently, our first objective in this chapter is to introduce language models of a system and present operations on languages that will be used in the next chapters.

2.1 Language notation

All the results of this section are rather general, we are not referring to a particular set. We begin by choosing a set A as the alphabet of our Language. We assume that set A is finite.

A sequence of elements taken out of this alphabet forms a "word" or "string". We shall use the term "string" in this text. A string consisting of no elements is called the empty string and is denoted by ε. Given a set A we denote $A_\varepsilon = A \cup \{\varepsilon\}$. The length of a string is the number of elements contained in it, counting multiple occurrences of the same element. If s is a string, we will denote its length by $|s|$. By convention, the length of the empty string ε is taken to be zero.

Definition 11 *A language defined over set A is a set of finite-length strings formed from elements in A.*

The key operation involved in building strings, and thus languages, from a set A is concatenation. The concatenation of two strings u and v is the string $u\,v$ consisting of the elements in u immediately followed by the elements in v. The empty string ε is the identity element of concatenation: $s\varepsilon = \varepsilon s = s$ for any string s.

Let us denote by A^* the set of all finite-length strings of elements of A, including the empty string ε; the $(\cdot)^*$ operation is called the Kleene-closure. Observe that the set A^* is countably infinity since it contains strings of arbitrarily long length. A language L over a set A is therefore a subset of A^*. In particular, $\varnothing$, A, and A^* are languages.

We conclude this section with some terminology about strings.
Let $s = s(1)\,s(2)\,\ldots\,s(n)$ be a string over the alphabet A, then:

- the i–th element of s is denoted by $s(i)$ and its last element by $s(end)$;
- A partition of a string $s \in A^*$ is a collection of strings $\{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k\}$, with $\hat{s}_i \in A^*$ for all $i = 1, \ldots, k$, such that their concatenation coincides with s, i.e. $s = \hat{s}_1\,\hat{s}_2\,\ldots\,\hat{s}_k$;
- a prefix of the string s is any string $s|_k = s(1)\,s(2)\,\ldots\,s(k)$ with $k \leq n$;
- a string w is a suffix of the string s if there exists a string p, prefix of s, such that $p\,w = s$.

We will use the notation s/p (read "s after p") to denote the suffix of s after its prefix p; if p is not a prefix of s, then s/p is not defined.

The following two definitions are new and instrumental for Chapter 5.

Definition 12 *Two strings $s_a, s_b \in A^*$ with $s_a \neq s_b$ are indistinguishable if s_a is a prefix of s_b or if s_b is a prefix of s_a; strings s_a, s_b are distinguishable, otherwise.*

Definition 13 *A set A of strings is distinguishable if it does not contain pairs of indistinguishable strings.*

2.2 Operation on languages

In this section we generalize to sets (i.e. Languages) the operations introduced in the previous section. The usual set operations, such as union, intersection, difference, and complement with respect to A^*, are applicable to languages since languages are sets.

- Let $L_a, L_b \subseteq A^*$ be two languages defined over the same alphabet A. The concatenation of L_a and L_b is the set $L_a\, L_b = \{a\, b \mid a \in L_a \wedge b \in L_b\}$ of concatenations of all ordered pairs of $L_a \times L_b$. The concatenation is an associative operation $L_a\, L_b\, L_c = (L_a\, L_b)\, L_c = L_a\, (L_b\, L_c)$.

- Let $L \subseteq A^*$ be a language over the alphabet A, then

$$L^* = \{\varepsilon\} \cup L \cup L\, L \cup L\, L\, L \cup \ldots$$

  An element of L^* is formed by the concatenation of a finite (but possibly arbitrarily large) number of elements of L. Note that the $(\cdot)^*$ operation is idempotent: $(L^*)^* = L^*$.

- Let $L \subseteq A^*$ be a language over the alphabet A, symbol

$$L^+ = L \cup L\, L \cup L\, L\, L \cup \ldots$$

  denotes the Kleene-plus of L, namely the Kleene-closure without the empty string ε.

- Let $L \subseteq A^*$ be a language over the alphabet A, the prefix-closure of a

language L is the set (language) consisting of all the prefixes of all the strings in L. More formally, we denote the prefix-closure as

$$\overline{L} = \{p \in A^* \mid \exists\, w \in A^* \wedge p\, w \in L\}$$

In general $L \subseteq \overline{L}$ and L is said to be prefix-closed when $L = \overline{L}$.

- Let $L \subseteq A^*$ be a language and $p \in \overline{L}$ be a string, over the alphabet A. We define the post-language of L after p, denoted by L/p, as the language

$$L/p = \{w \in A^* \mid p\, w \in L\}$$

By definition, if $p \notin \overline{L}$ then $L/p = \varnothing$.

2.3 Languages and systems

The task of this section is to explain how to translate the information encoded in the tuple of a system into a language and vice versa.

One way to connect languages and systems arises by inspecting the state transition diagram of a system. Consider all the directed paths that can be followed in the state transition diagram, starting at the initial state; consider among these all the paths that end in a marked state. This leads to the notions of (Cassandras & Lafortune, 1999) of the input languages and marked input languages generated by a system.

Definition 14 *The input language generated by system* $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$ *is*

$$\mathcal{L}^u(S) := \{w \in U^* \mid f(x_0, w) \textit{ is defined}\}$$

The input language generated by system S can be seen as the collection of those input runs strictly associated to those state run of S starting from the initial state.

Definition 15 *The marked input language generated by system* $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$ *is*

$$\mathcal{L}_m^u(S) := \{w \in \mathcal{L}(S) \mid f(x_0, w) \in X_m\}$$

The marked input language of system S can be seen as the collection of those input runs strictly associated to those state runs of S starting from the initial state and ending in a marked state.

From the definitions above it is clear that we are referring to deterministic systems (Definition 1) because it is impossible, for a language as in Definition 14, to account the nondeterminism of a systems. Hence, by convention, the system associated to a language is always deterministic.

For the sake of clarity we point out that with the only information encoded in the input language we cannot represent the entire system. In fact, the previous language does not consider the outputs. In order to fully represent a system we need to introduce the output language and the marked output language.

Definition 16 *The output language generated by system* $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$*, denoted by* $\mathcal{L}^y(S)$*, is the collection of the output runs associated to any state runs of* S*.*

Definition 17 *The marked output language generated by system* $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$*, denoted by* $\mathcal{L}_m^y(S)$*, is the collection of the output runs associated to those state runs of* S *ending in a marked state.*

As for the input language, in order to correctly translate the information of the

system, we need a constraint on the system. We require the system to be output deterministic (ref. Definition 10).

At this point it is easy to understand what the last possible alphabet over a system S is: the set of states X. The language generated by the set of states is generally used to define the specification to enforce over a system to control and we do not need to have a deterministic system.

Due to control needs we are normally interested to languages that can be represented (generated and marked) by symbolic systems. Then we give the following

Definition 18 *A language L over a finite set U is regular if there exists a symbolic system S with input set U such that $L = \mathcal{L}_m^u(S)$.*

For the sake of completeness we recall the following example of a non regular language.

Example 5 *Consider the language $L = \{a^n\, b^n \mid n \in \mathbb{N}\}$ over the set $U = \{a, b\}$. This language cannot be represented by a symbolic system. Indeed, we observe that a marked state must be reached after exactly the same number of b events as that of a events that started the string. Therefore, the system must have memory of how many a events occurred when it starts allowing b events, at which point it must also count the number of occurrences of b events; this is necessary in order to allow the correct number of b events before entering a marked state.*

For future purposes we recall that operation of union, subtraction, complement, concatenation, Kleene-closure and prefix closure over regular languages are regular languages. More formally:

Theorem 1 *Let L_1 and L_2 be two regular languages over alphabet U. The following languages are also regular:*

- $\overline{L_i}$, *for* $i = 1, 2$;
- L_i^*, *for* $i = 1, 2$;
- $L_i^c = U^* \setminus L_i$, *for* $i = 1, 2$;
- $L_1 \cup L_2$;
- $L_1 \setminus L_2$ *and* $L_2 \setminus L_1$;
- $L_1 \cap L_2$;
- $L_1\, L_2$ *and* $L_2\, L_1$.

At the end of this chapter we show how to convert a system such that the input language becomes the output language. Given a regular language L_Q over the alphabet $\mathbf{L}$ we first need to encode the language in a system. Hence, since language L_Q is regular there exists a symbolic system

$$S'_Q = (X'_Q, x'_{Q,0}, \mathbf{L}, f'_Q, X'_{Q,m}, \Gamma'_Q, Y'_Q, H'_Q),$$

such that its marked input language coincides with L_Q,

$$\mathcal{L}_m^u(S'_Q) = L_Q.$$

Note that in the definition above the output function can be chosen arbitrarily because it plays no role in ensuring $\mathcal{L}_m^u(S'_Q) = L_Q$. Without loss of generality, S'_Q can be chosen as deterministic and nonblocking, see e.g. (Cassandras & Lafortune, 1999).

For future purposes it is necessary to have that the output language coincide with L_Q. Hence, we introduce the dual symbolic system S_Q of system S'_Q, where states of S_Q are transitions of S'_Q and vice versa.

Definition 19 *Given system S'_Q, define the dual system*

$$S_Q = (X_Q, X_{Q,0}, U_Q, f_Q, X_{Q,m}, \Gamma_Q, \mathbf{L}, H_Q) \tag{2.1}$$

where:

- *X_Q is the collection of distinct triplets (x, u, x^+) for which $x^+ = f'_Q(x, u)$, namely transitions of S'_Q;*
- *$X_{Q,0}$ is the collection of those states (x, u, x^+) in X_Q for which $x = x'_{Q,0}$;*
- *$U_Q = \{u_Q\}$, where u_Q is a dummy input;*
- *$f_Q : X_Q \times \{u_Q\} \to 2^{X_Q}$ is defined as follows: for any pairs (x_1, u_1, x_2), $(x_3, u_2, x_4) \in X_Q$ we set*

$$(x_3, u_2, x_4) \in f_Q((x_1, u_1, x_2), u_Q)$$

  if $x_2 = x_3$;
- *$X_{Q,m}$ is the collection of those states (x, u, x^+) in X_Q for which $x^+ \in X'_{Q,m}$;*
- *$H_Q : X_Q \to Y_Q$ is defined by*

$$H_Q(x, u, x^+) = u,$$

  for any (x, u, x^+) in X_Q.

From the definitions above, it follows that $\mathcal{L}^y(S_Q) = \mathcal{L}^u(S'_Q) \setminus \{\varepsilon\}$ and $\mathcal{L}^y_m(S_Q) = \mathcal{L}^u_m(S'_Q) \setminus \{\varepsilon\}$. Moreover, S_Q is symbolic, accessible and nonblocking. Since u_Q plays no role, for ease of notation we denote a transition of S_Q as $x^+_Q \in f_Q(x_Q)$ instead $x^+_Q \in f_Q(x_Q, u_Q)$, with $x_Q, x^+_Q \in X_Q$.

For the sake of clarity we give the following example

Example 6 *Consider the alphabet* $\mathbf{L} = \{a, b, c\}$ *and the language* $L_Q = \{a\, b^n\, c \mid n \in \mathbb{N}_0\}$ *over it. We choose* $X'_Q = \{x_0, x_1, x_2\}$, $x'_{Q,0} = x_0$ *and* Y'_Q *is not represented because it plays no role. The resulting deterministic symbolic system that marks* L_Q *is depicted in Figure 2.1.*

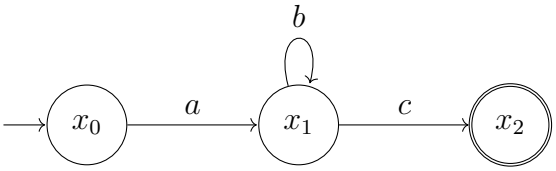

Figure 2.1: System S'_Q of Example 6.

Now we compute the dual system S_Q *but, for ease of notation, we rename the states in* X_Q *as* $x_a = (x_0, a, x_1)$, $x_b = (x_1, b, x_1)$ *and* $x_c = (x_1, c, x_2)$. *Hence,* $X_{Q,0} = \{x_a\}$ *and* $X_{Q,m} = \{x_c\}$. *The resulting symbolic system is depicted in Figure 2.2.*

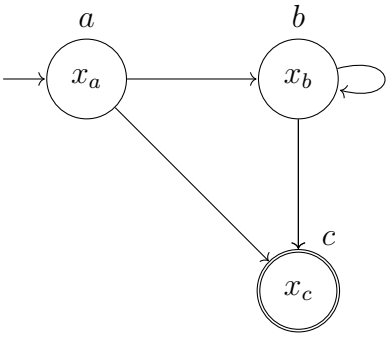

Figure 2.2: System S_Q of Example 6.

3 | Operations on Systems

In this chapter we consider operations that alter the state transition diagram of a system. All systems considered in this chapter are deterministic. In the sequel we will extend these operators to the nondeterministic case.

3.1 Unary operations

In the previous section we have seen that it is possible to associate a system to a language and vice versa. But this operation is not reversible, in general.

Example 7 *Consider system S as in Figure 3.1, we do not care of outputs at this stage.*

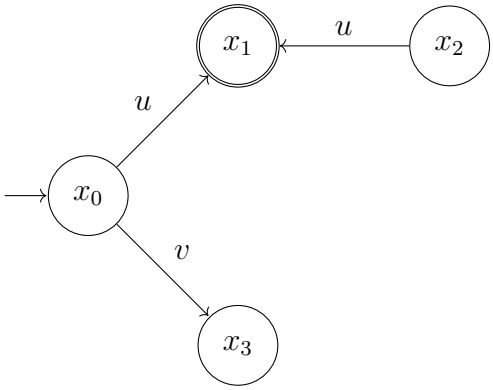

Figure 3.1: System S of Example 7.

The input language of S is $\mathcal{L}^u(S) = \{u, v\}$ and the marked input language is $\mathcal{L}^u_m(S) = \{u\}$. Now, we try to build the same system with the only information of the languages. The resulting system is represented in Figure 3.2.

We point out that since in $\mathcal{L}^u(S)$ and $\mathcal{L}^u_m(S)$ the information of the set of states of S is not encoded, system S' has a different set of states. For our sake, at this stage, the set of states does not matter, we focus on the graph.

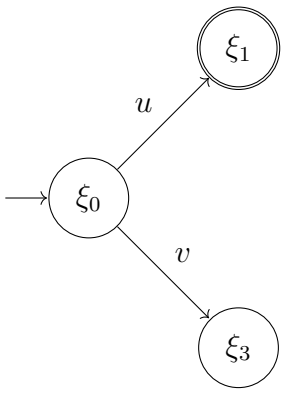

Figure 3.2: System S' based on the information of languages of S.

Indeed, it is easy to see that $\mathcal{L}^u(S) = \mathcal{L}^u(S')$ and $\mathcal{L}^u_m(S) = \mathcal{L}^u_m(S')$ but the graph of S is different from the graph of S', i.e. we have a missing transition in S'.

State x_2 is said to be not accessible, namely there is no state run ending in the state x_2.

From the example above, we see that we can delete from S all the states that are not accessible or reachable from a state run in S, without affecting the languages generated and marked by S. When we "delete" a state, we also delete all the transitions that are attached to that state. We will denote this operation by $Ac(S)$, where Ac stands for taking the "accessible" part. Formally

Definition 20 *Given a system $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$ as in Definition 1, the accessible part of S (denoted $Ac(S)$) is*

$$Ac(S) := (X_{ac}, x_0, U, f_{ac}, X_{ac,m}, \Gamma_{ac}, Y, H_{ac})$$

where

- $X_{ac} := \{x \in X \mid f(x_0, s) = x \text{ for some } s \in U^*\}$;
- $f_{ac}|_{X_{ac} \times U \to X_{ac}}$, *domain and codomain of f restriction to X_{ac};*

- $X_{ac,m} := X_m \cap X_{ac}$;
- $\Gamma_{ac}|_{X_{ac}}$;
- $H_{ac}|_{X_{ac}}$.

Clearly, the Ac operation has no effect on $\mathcal{L}^u(S)$ (resp. $\mathcal{L}^y(S)$) and $\mathcal{L}^u_m(S)$ (resp. $\mathcal{L}^y_m(S)$). Thus, we say that a system S is accessible if $Ac(S) = S$.

We have seen how to delete all those states that are not reachable from the initial state. Now we want to delete all those states that cannot reach a marked state. A state x of S is said to be coaccessible to X_m, or simply coaccessible, if there is a path in the state transition diagram of S from state x to a marked state. We denote the operation of deleting all the states of S that are not coaccessible by $CoAc(S)$, where $CoAc$ stands for taking the "coaccessible" part. Formally

Definition 21 *Given a system $S = (X, x_0, U, f, X_m, \Gamma, Y, H)$ as in Definition 1, the coaccessible part of S (denoted $CoAc(S)$) is*

$$CoAc(S) := (X_{coac}, x_{coac,0}, U, f_{coac}, X_m, \Gamma_{coac}, Y, H_{coac})$$

where

- $X_{coac} := \{x \in X \mid f(x, s) \in X_m \text{ for some } s \in U^*\}$;
- $x_{coac,0} := \begin{cases} x_0, & \text{if } x_0 \in X_{coac} \\ undefined & otherwise \end{cases}$;
- $f_{coac}|_{X_{coac} \times U \to X_{coac}}$, *domain and codomain of f restriction to X_{coac};*
- $\Gamma_{coac}|_{X_{coac}}$;
- $H_{coac}|_{X_{coac}}$.

Example 7 **(cont.)** Consider system S as in Figure 3.1, the coaccessible part of S is depicted in Figure 3.3.

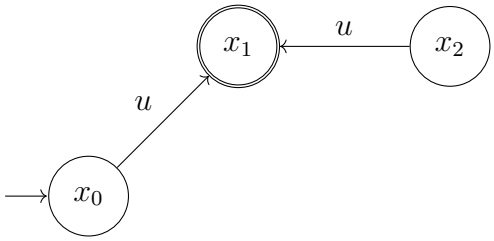

Figure 3.3: System $CoAc(S)$ of Example 7.

The input language of $CoAc(S)$ is $\mathcal{L}^u(CoAc(S)) = \{u\}$ and the marked input language is $\mathcal{L}^u_m(CoAc(S)) = \{u\}$, the same of S.

The $CoAc$ operation may shrink $\mathcal{L}^u(S)$ (or $\mathcal{L}^y(S)$), since we may be deleting states that are accessible; however, the $CoAc$ operation does not affect $\mathcal{L}^u_m(S)$ ($\mathcal{L}^y_m(S)$). It is easy to see that whenever a system S is coaccessible $CoAc(S) = S$ then the nonblocking property is matched; in this case $\mathcal{L}^u(S) = \overline{\mathcal{L}^u_m(S)}$.

A system that is both accessible and coaccessible is said to be trim. We define the $Trim$ operator as

Definition 22 *Given system S as in Definition 1 we define*

$$Trim(S) := Ac(CoAc(S)) = CoAc(Ac(S)).$$

It is easy to verify that operators Ac and $CoAc$ commute in the case of deterministic systems.

3.2 Composition operations

We start this section by introducing the following

Definition 23 *Let $S_i = (X_i, X_{i,0}, U_i, f_i, X_{i,m}, \Gamma_i, Y_i, H_i)$ be two nondeterministic systems with $i = 1, 2$, we say that S_1 is a subsystem of S_2, denoted $S_1 \sqsubseteq S_2$, if*

- $X_1 \subseteq X_2$;
- $X_{1,0} \subseteq X_{2,0}$;
- $U_1 \subseteq U_2$;
- $f_1(x, u) \subseteq f_2(x, u)$ *for all* $x \in X_1$ *and* $u \in U_1$;
- $X_{1,m} \subseteq X_{2,m}$;
- $\Gamma_1(x) \subseteq \Gamma_2(x)$ *for all* $x \in X_1$;
- $Y_1 \subseteq Y_2$;
- $H_1(x) = H_2(x)$ *for all* $x \in X_1$;

Binary operator $\sqsubseteq$ is a pre–order on the set of systems because it enjoys

- **reflexivity property:** $S \sqsubseteq S$ for any system S;
- **transitivity property:** $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_3$ implies $S_1 \sqsubseteq S_3$ for any systems S_1, S_2 and S_3.

We now define two operations on systems: product, denoted by $\times$, and parallel composition, denoted by $||$. For simplicity, we present these operations for two deterministic systems. Nondeterministic systems can be composed using the same rules for the transition map. They are easily generalized to the composition of a set of systems using the associativity properties.

We can think of the two operations as two types of interconnections of systems with different input sets, in general. As we will see, the key difference between these two operations pertain to how private inputs, i.e., inputs that are not in common, are handled.

Definition 24 *Given two systems* $S_i = (X_i, x_{i,0}, U_i, f_i, X_{i,m}, \Gamma_i, Y_i, H_i)$ *with* $i = 1, 2$*, the product of* S_1 *and* S_2 *is the system*

$$S_1 \times S_2 := Ac(X_1 \times X_2, x_{1,0} \times x_{2,0}, U_1 \cup U_2, f_\times, X_{1,m} \times X_{2,m}, \Gamma_\times, Y_1, H_\times)$$

where

- $f_\times((x_1, x_2), u) := \begin{cases} (f_1(x_1, u), f_2(x_2, u)) & \text{if } u \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ undefined & otherwise \end{cases}$;
- $\Gamma_\times(x_1, x_2) := \Gamma_1(x_1) \cap \Gamma_2(x_2)$;
- $H_\times(x_1, x_2) := H(x_1)$.

In the product definition the outputs have no role so we defined, by convention, the output of the first system. Later on we will extend this concept in order to include the outputs.

In the product operation, the transitions of the two systems must always be synchronized on a common input, that is an input in $U_1 \cap U_2$. An input occurs if and only if it occurs in both systems. The states of $S_1 \times S_2$ are denoted by pairs, where the first component is the (current) state of S_1 and the second component is the (current) state of S_2. It is easily verified that

$$\begin{aligned} \mathcal{L}^u(S_1 \times S_2) &= \mathcal{L}^u(S_1) \cap \mathcal{L}^u(S_2) \\ \mathcal{L}^u_m(S_1 \times S_2) &= \mathcal{L}^u_m(S_1) \cap \mathcal{L}^u_m(S_2) \end{aligned}$$

This shows that the intersection of two languages can be "implemented" by doing the product of their system representations. If $U_1 \cap U_2 = \varnothing$, then $\mathcal{L}^u(S_1 \times S_2) = \{\varepsilon\}$; $\mathcal{L}^u_m(S_1 \times S_2)$ will be either $\varnothing$ or $\{\varepsilon\}$, depending on the marking status of the initial state $(x_{1,0}, x_{2,0})$. The input set of $S_1 \times S_2$ is defined to be $U_1 \cup U_2$, in order to

record the original input sets in case these are needed later on; we will comment further on this issue when discussing parallel composition next. However, the active inputs of $S_1 \times S_2$ will necessarily be in $U_1 \cap U_2$. In fact, not all inputs in $U_1 \cap U_2$ need to be active in $S_1 \times S_2$, as this depends on the joint transition function $f_\times$.

Properties of product composition.

1. Product composition is commutative up to a reordering of the state components.

2. Product composition is associative.

Composition by product is restrictive as it only allows transitions on common inputs. In general, when modeling systems composed of interacting components, the input set of each component includes private inputs that pertain to its own internal behaviour and common inputs that are shared with other systems and capture the coupling among the respective system components. The standard way of building models of entire systems from models of individual system components is by parallel composition.

Definition 25 *Given two systems* $S_i = (X_i, x_{i,0}, U_i, f_i, X_{i,m}, \Gamma_i, Y_i, H_i)$ *with* $i = 1, 2$*, the parallel composition of* S_1 *and* S_2 *is the system*

$$S_1||S_2 := Ac(X_1 \times X_2, x_{1,0} \times x_{2,0}, U_1 \cup U_2, f_{||}, X_{1,m} \times X_{2,m}, \Gamma_{||}, Y_1, H_{||})$$

where

- $f_{||}((x_1, x_2), u) := \begin{cases} (f_1(x_1, u), f_2(x_2, u)) & \textit{if } u \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, u), x_2) & \textit{if } u \in \Gamma_1(x_1) \setminus U_2 \\ (x_1, f_2(x_2, u)) & \textit{if } u \in \Gamma_2(x_2) \setminus U_1 \\ undefined & otherwise \end{cases}$;
- $\Gamma_{||}(x_1, x_2) := (\Gamma_1(x_1) \cap \Gamma_2(x_2)) \cup (\Gamma_1(x_1) \setminus U_2) \cup (\Gamma_2(x_2) \setminus U_1)$;
- $H_{||}(x_1, x_2) := H(x_1)$.

Even in this case the outputs have no role so we defined, by convention, the output of the first system.

In the parallel composition, a common input, which is an input in $\Gamma_1(x_1) \cap \Gamma_2(x_2)$, can only be executed if the two systems both execute it simultaneously. Thus, the two systems are "synchronized" on the common input. The private inputs, that is, those in $(U_1 \setminus U_2) \cup (U_2 \setminus U_1)$, are not subjected to such a constraint and can be executed whenever possible. In this kind of interconnection, a system can execute its private inputs without the participation of the other one; however, a common input is enabled if both systems can execute it. If $U_1 = U_2$, then the parallel composition reduces to the product, since all transitions are forced to be synchronized.

Properties of parallel composition.

1. Parallel composition is commutative up to a reordering of the state components.

2. Parallel composition is associative.

We introduced the class of nondeterministic systems, which differ from deterministic ones by allowing the codomain of f to be 2^X, the power set of the state

space of the system, and also allowing multiple initial states. We show now how to convert a nondeterministic system in a deterministic one preserving the same languages. The state space of the equivalent deterministic system will be a subset of the power set of the state space of the nondeterministic one. This means that if the nondeterministic system is finite-state, then the equivalent deterministic one will also be finite-state. We shall call the resulting equivalent deterministic system the observer corresponding to the original nondeterministic system.

Before defining the algorithm of the observer we need to introduce the function $\varepsilon R(\cdot) : 2^X \to 2^X$, the ε-reach of states in $A \subseteq X$:

$$\varepsilon R(A) := f^{ext}(A, \varepsilon)$$

where we recall from Section 1.2 the definition of f^{ext}, the extension of transition function of nondeterministic systems in Definition 7.

Definition 26 *Let $S = (X, X_0, U \cup \{\varepsilon\}, f, X_m, \Gamma, Y, H)$ be a nondeterministic system. Then $Obs(S) = (X_{Obs}, X_{Obs,0}, U, f_{Obs}, X_{Obs,m}, \Gamma_{Obs})$ and it is built as follows.*

1. *Set $X_{Obs,0} = \varepsilon R(X_0)$ and $X_{Obs} = \{X_{Obs,0}\}$.*
2. *For each $A \in X_{Obs}$ and $u \in U$, define*

$$f_{Obs}(A, u) := \varepsilon R(f^{ext}(A, u))$$

   and set $X_{Obs} = X_{Obs} \cup \{f_{Obs}(A, u)\}$.
3. *Repeat step 2. until the entire accessible part of $Obs(S)$ has been constructed.*
4. $X_{Obs,m} := \{A \in X_{Obs} \mid A \cap X_m \neq \varnothing\}$.

In this definition of observer no information about the output is given. In the following we will extend it in order to include the information of the output.

Important properties of $Obs(S)$ are:

- $Obs(S)$ is a deterministic system;
- $\mathcal{L}^u(Obs(S)) = \mathcal{L}^u(S)$;
- $\mathcal{L}^u_m(Obs(S)) = \mathcal{L}^u_m(S)$.

3.3 Equivalence of systems

We established in the preceding section that nondeterministic systems are language equivalent to deterministic systems. Any nondeterministic system can be transformed to a language-equivalent deterministic one.

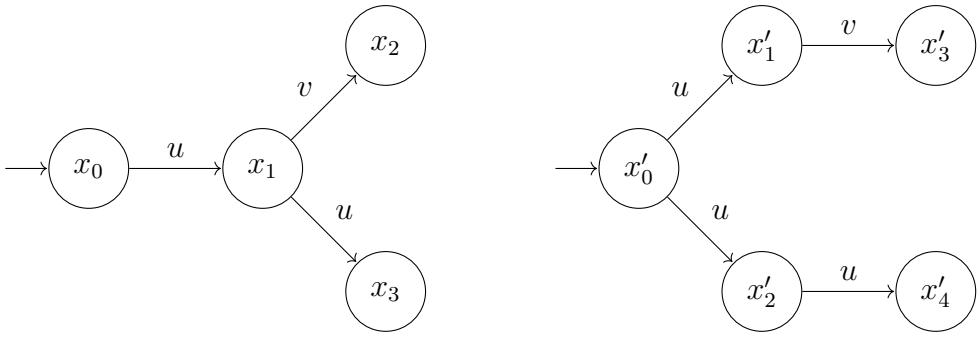

Figure 3.4: Tow systems with the same language.

A widely used notion of equivalence is that of bisimulation. Bisimulation equivalence stipulates that any pair of states reached after a given string of inputs should have the same future behaviour in terms of post-language. In the case of deterministic systems, bisimulation reduces to language equivalence. This notion of equivalence is formalized by introducing the notion of bisimulation relation with same set of inputs U. For the sake of clarity we formalize this concept.

Definition 27 *Given two systems $S_i = (X_i, X_{i,0}, U, f_i, X_{i,m}, \Gamma_i, Y_i, H_i)$ with $i = 1, 2$, we say that S_1 and S_2 are bisimilar if there exists a binary relation $R \subseteq X_1 \times X_2$ such that*

B1. *a. for each $x_1 \in X_1$ there exists $x_2 \in X_2$ such that $(x_1, x_2) \in R$;*

b. for each $x_2 \in X_2$ there exists $x_1 \in X_1$ such that $(x_1, x_2) \in R$;

B2. *a. if $(x_1, x_2) \in R$, $u \in U$ and $x_1' \in f_1(x_1, u)$, then there exists $x_2' \in f_2(x_2, u)$ such that $(x_1', x_2') \in R$;*

b. if $(x_1, x_2) \in R$, $u \in U$ and $x_2' \in f_2(x_2, u)$, then there exists $x_1' \in f_1(x_1, u)$ such that $(x_1', x_2') \in R$;

B3. *$(x_1, x_2) \in R$ implies $x_1 \in X_{1,m}$ if and only if $x_2 \in X_{2,m}$.*

Bisimilarity is an equivalence relation among systems since it is symmetric, reflexive, and transitive.

This relation is extensively used in the simulation and control of continuous systems. For the sake of completeness we present some other notions of bisimulation.

Definition 28 *Given two metric systems $S_i = (X_i, X_{i,0}, U_i, f_i, X_{i,m}, \Gamma_i, Y, H_i)$ with $i = 1, 2$ and same metric d, and given an accuracy $\mu \in \mathbb{R}_0^+$. We say that S_1 and S_2 are μ-approximate bisimilar if there exists a binary relation $R \subseteq X_1 \times X_2$ such that*

B1. *a. for each $x_1 \in X_{1,0}$ and $x_2 \in X_2$ such that $(x_1, x_2) \in R$, it holds that $x_2 \in X_{2,0}$;*

b. for each $x_2 \in X_{2,0}$ and $x_1 \in X_1$ such that $(x_1, x_2) \in R$, it holds that $x_1 \in X_{1,0}$;

B2. a. *if* $(x_1, x_2) \in R$, $u_1 \in U_1$ *and* $x_1' \in f_1(x_1, u_1)$, *then there exists* $u_2 \in U_2$ *and* $x_2' \in f_2(x_2, u_2)$ *such that* $(x_1', x_2') \in R$;

b. *if* $(x_1, x_2) \in R$, $u_2 \in U_2$ *and* $x_2' \in f_2(x_2, u_2)$, *then there exists* $u_1 \in U_1$ *and* $x_1' \in f_1(x_1, u_1)$ *such that* $(x_1', x_2') \in R$;

B3. $(x_1, x_2) \in R$, $d(H_1(x_1), H_2(x_2)) \leq \mu$.

This notion requires trajectories of approximately bisimilar systems to be step-by-step close to each other up to a desired and given accuracy.

Definition 29 *Given two metric systems* $S_i = (X_i, X_{i,0}, U, f_i, X_{i,m}, \Gamma_i, Y, H_i)$ *with* $i = 1, 2$ *and same metric* d, *and given an accuracy* $\mu \in \mathbb{R}_0^+$. *We say that* S_1 *and* S_2 *are strong* μ*-approximate bisimilar if there exists a binary relation* $R \subseteq X_1 \times X_2$ *such that*

B1. a. *for each* $x_1 \in X_{1,0}$ *and* $x_2 \in X_2$ *such that* $(x_1, x_2) \in R$, *it holds that* $x_2 \in X_{2,0}$;

b. *for each* $x_2 \in X_{2,0}$ *and* $x_1 \in X_1$ *such that* $(x_1, x_2) \in R$, *it holds that* $x_1 \in X_{1,0}$;

B2. a. *if* $(x_1, x_2) \in R$, $u \in U$ *and* $x_1' \in f_1(x_1, u)$, *then there exists* $x_2' \in f_2(x_2, u)$ *such that* $(x_1', x_2') \in R$;

b. *if* $(x_1, x_2) \in R$, $u \in U$ *and* $x_2' \in f_2(x_2, u)$, *then there exists* $x_1' \in f_1(x_1, u)$ *such that* $(x_1', x_2') \in R$;

B3. $(x_1, x_2) \in R$, $d(H_1(x_1), H_2(x_2)) \leq \mu$.

In order to highlight the difference among the bisimulation and the (strong) μ-approximate bisimulation we propose the following examples.

Example 8 *Consider systems S_1 and S_2 depicted in Figure 3.5 and 3.6, respectively.*

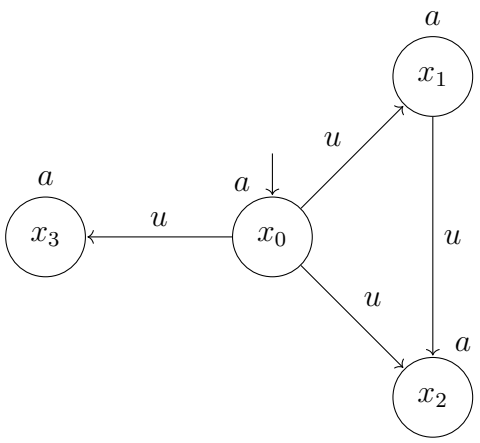

Figure 3.5: System S_1 of Example 8.

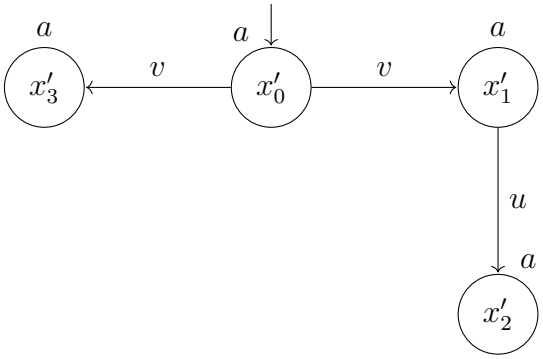

Figure 3.6: System S_2 of Example 8.

The two systems are not bisimilar; it is easy to see that system S_2 has evolutions with an input that is not present in S_1, i.e. input v. For the same reason the systems are not strong μ-approximate bisimilar. At this stage we do not care about parameter μ because all the states, of both systems, have the same output a.

Instead, systems S_1 and S_2 are μ-approximate bisimilar. In fact, there exists a bisimulation relation R for which the two systems are μ-approximate bisimilar

for all $\mu \in \mathbb{R}^+$. Let's see this relation and prove our claim.

$$R = \{(x_0, x'_0), (x_1, x'_1), (x_2, x'_2), (x_2, x'_3), (x_3, x'_3)\}$$

Condition B1 of Definition 28 is trivially true, x_0 and x'_0 are the only initial states of systems S_1 and S_2, respectively and hence the pair (x_0, x'_0) is in R. Moreover, there is no other pair of states in R that contains an initial state.

Let's verify condition B2 of Definition 28:

- *consider pair (x_0, x'_0), for each transition from x_0 there is a transition from x'_0 such that the pair of the reached states is in the relation R, and vice versa. The pairs above are*

$$(x_1, x'_1) \quad (x_2, x'_3) \quad (x_3, x'_3).$$

- *Now, consider pair (x_1, x'_1), from both the states there is only one transition and the pair of reached states is in R, i.e. (x_2, x'_2).*
- *From the other pairs in R there are no evolutions.*

Supposing we replace the input labels v with the input label u we get that the two systems are bisimilar and strong μ-approximate bisimilar too. Moreover, the bisimulation relation R is the same.

Example 9 *Consider systems S_1 and S_2 depicted in Figure 3.7.*

The two systems above are bisimilar with a bisimulation relation

$$R = \{(x_0, x'_0), (x_1, x'_1), (x_1, x'_2), (x_2, x'_1), (x_2, x'_2)\}.$$

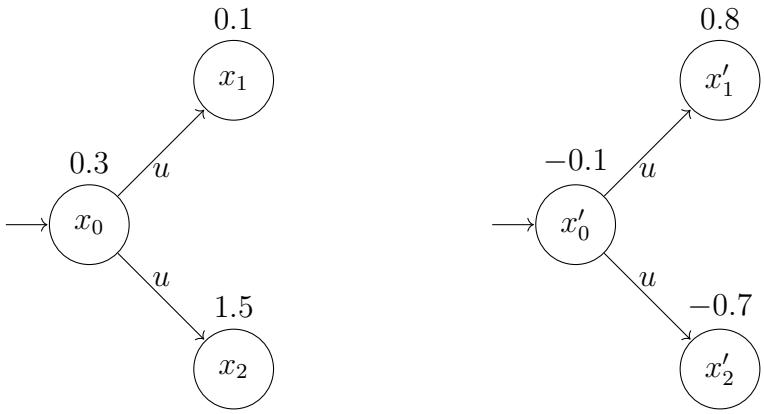

Figure 3.7: System S_1 (on the left) and system S_2 (on the right) of Example 9.

We point out that this is one of the possible bisimulation relation, in fact also

$$R = \{(x_0, x'_0), (x_1, x'_1), (x_2, x'_2)\}$$

and

$$R = \{(x_0, x'_0), (x_1, x'_2), (x_2, x'_1)\}$$

and so on are fine with this example.

With the other bisimulations we get different results depending on parameter μ.

- *If $\mu < 0.8$ the bisimulation relation is empty;*
- *if $0.8 \leq \mu < 2.2$ the bisimulation relation is*

$$R = \{(x_0, x'_0), (x_1, x'_2), (x_2, x'_1)\};$$

- *if $\mu \geq 2.2$ we can have several bisimulation relation R as for the bisimulation of Definition 27.*

Since in this example there is only one input we have no differences between μ-approximate bisimulation and strong μ-approximate bisimulation.

4 | Symbolic control of nonlinear systems with dynamic specifications

This chapter starts with some recalls regarding the symbolic control design. Then, it details the new control problem of reconfiguring on-the-fly the controller according with an unexpected change of the specification. The example shows the time computational benefit with respect to other traditional approaches. The results of this chapter are based on the works (Masciulli & Pola, 2021, 2020).

4.1 Introduction

In this chapter we consider a control problem where:

- The plant is described by a continuous–time incrementally stable nonlinear system.
- The controller is described by a symbolic system that then, is easily implementable in digital devices.
- The specifications are expressed as regular languages that, as also stressed in (Tabuada, 2009), are relevant in the control design of many concrete applications.

We consider a situation where the system does not know all the specification a priori, but has only the knowledge of a first part of it; then, at some time the system will obtain a second specification for which the controller needs to reconfigure. We propose efficient algorithms for the design of controllers in this framework. An analysis, included in the chapter, shows that our approach presents gain in terms of time computational complexity with respect to other traditional

approaches. On the other hand, space computational complexity needed in our approach and the traditional ones, are comparable.

Central to this approach is the construction of systems that approximate purely continuous or hybrid plants. This abstraction consists in a symbolic system where each state corresponds to an aggregate of continuous/hybrid states and each label to an aggregate of continuous/hybrid inputs.

The literature on symbolic systems and on their use to verification and control purposes is very rich, see e.g. (Tabuada, 2009; Belta, Yordanov, & Aydin Gol, 2017; Girard & Pappas, 2011; Pola & Di Benedetto, 2019) and the references therein. To the best of our knowledge, most of current results always consider specifications that are fixed a priori. However, there are some applications where the system does not know the whole specification a priori.

4.2 Plant and control system

The control scheme we consider in this chapter consists of a plant P, a controller C, and a Zero order Holder (ZoH). The plant P is described by a continuous–time nonlinear control system:

$$P : \begin{cases} \dot{x}(t) = F(x(t), u(t)), \\ x(t) \in \mathbf{X} \subseteq \mathbb{R}^n, \\ x(0) \in \mathbf{X}_0 \subseteq \mathbf{X}, \\ u(t) \in \mathbf{U} \subseteq \mathbb{R}^m, t \in \mathbb{R}_0^+, \end{cases} \tag{4.1}$$

where: $x(t)$ and $u(t)$ denote, respectively, the state and the control input at time $t \in \mathbb{R}_0^+$; $\mathbf{X}$ is the state space; $\mathbf{X}_0$ is the set of initial states; $\mathbf{U}$ is the input set; for the function F we suppose the following Lipschitz assumption holds.

Assumption 1 *Consider the continuous function $F : \mathbf{X} \times \mathbf{U} \to \mathbb{R}^n$. For every compact set $K \subseteq \mathbf{X}$, there exists a constant $\kappa \in \mathbb{R}^+$ such that*

$$\|F(x,u) - F(x',u)\| \leq \kappa \|x - x'\|,$$

for all $x, x' \in K$ and all $u \in \mathbf{U}$.

Since we are interested in controlling the plant P through a digital and quantized controller, we assume that the set $\mathbf{U}$ is finite and denote by $\mathcal{U}$ the set of piecewise constant functions u from $\mathbb{R}_0^+$ to $\mathbf{U}$ such that $u(t) = u(k\tau)$ for all $t \in [k\tau, (k+1)\tau[$ and $k \in \mathbb{N}_0$, where $\tau \in \mathbb{R}^+$ is the clock period of the microprocessor implementing the controller; moreover, symbol $\mathcal{U}_\tau$ denotes the finite set of constant functions from $[0, \tau[$ to $\mathbf{U}$. Given $t_f \in \mathbb{R}^+$, a function $x : [0, t_f] \to \mathbb{R}^n$ is said to be a *state trajectory* of P if there exists $u \in \mathcal{U}$ satisfying $\dot{x}(t) = F(x(t), u(t))$, for almost all $t \in [0, t_f]$. We also denote by $\mathbf{x}(t, x_0, u)$ the state reached at time t under the input u from initial condition x_0; this state is uniquely determined, since the assumptions on F ensure existence and uniqueness of trajectories.

Controller C is given in the form of a deterministic symbolic system in the sense of Definition 1 (but with possibly more than one initial state), as follows:

$$C = (X_c, X_{c,0}, \mathbf{U}, f_c, X_{c,m}, Y_c, H_c), \tag{4.2}$$

where X_c is the set of states, $X_{c,0} \subseteq X_c$ is the set of initial states, $\mathbf{U}$ is the set of inputs, $f_c : X_c \times \mathbf{U} \to X_c$ is the transition function, $X_{c,m} \subseteq X_c$ is the set of marked states, Y_c is the set of outputs, $H_c : X_c \to Y_c$ is the output function. Given $k_f \in \mathbb{N}_0$, a sequence $x_c = \{x_{c,k}\}_{k=0,\ldots,k_f}$ of elements in X_c is said to be a *state trajectory* of C if for all $k \in [0; k_f - 1]$ there exists $u_k \in \mathbf{U}$ such that $x_{c,k+1} = f_c(x_{c,k}, u_k)$. Since the mathematical model of C is the same as the one

typically used in modeling software and hardware in microprocessors, controller C is easily implementable in digital devices.

A Zero order Holder (ZoH) block is placed in between P and C and is described by:

$$u(t) = u_k,\ t \in [k\tau, (k+1)\tau[,\ k \in \mathbb{N}_0. \tag{4.3}$$

The interconnection between the plant P and the controller C is formalized as follows:

Definition 30 *Given plant P in (4.1) and controller C in (4.2), we denote by P^C the system obtained by the interconnection between P and C, which is given by the collection of all pairs of state trajectories (x, x_c) satisfying (4.1), (4.2) and (4.3), where $(x(0), x_{c,0}) \in \mathbf{X}_0 \times X_{c,0}$.*

From the definition of P^C above, system C acts on plant P as an open–loop controller. As discussed in e.g. (Pola & Di Benedetto, 2019), when deterministic control systems are considered, as in this chapter, open–loop controllers are general enough to enforce regular language specifications; when nondeterministic control systems are considered, closed–loop controllers need to be used because they can guarantee a control action that is robust with respect to inherent nondeterminism.

4.3 Symbolic control design

This section recalls the results of (Pola & Di Benedetto, 2019; Pola, Di Benedetto, & Borri, 2019).

Consider a finite subset $\mathcal{Y}$ of the state space $\mathbf{X}$ of P and a specification expressed as a regular language $L \subseteq \mathcal{Y}^*$, where we recall $\mathcal{Y}^*$ is the Kleene closure of $\mathcal{Y}$. We start by recalling a rather basic problem in symbolic control design:

Problem 1 *Given the plant P in (4.1), the sampling time (clock period) $\tau \in \mathbb{R}^+$, the specification L and a desired accuracy $\theta \in \mathbb{R}^+$, find a controller C as in (4.2) and a relation $\mathcal{R}^0 \subseteq \mathbf{X}_0 \times X_{c,0}$ such that for any trajectory (x, x_c) of P^C with pair of initial states $(x(0), x_{c,0}) \in \mathcal{R}^0$, there exist $k_f \in \mathbb{N}_0$ and a word $q_0 q_1 ... q_{k_f} \in L$ such that for all $k \in [0; k_f]$*

$$\|x(k\tau) - q_k\| \leq \theta. \tag{4.4}$$

We point out that in the problem formulation above the specification is required to be met only at times $t = k\tau$, i.e. at times that are multiple of the sampling time τ; hence no condition is imposed in the inter-sampling times $]k\tau, (k+1)\tau[$, for all $k \in [0; k_f]$. Specifications involving also inter–sampling times are not considered here for the sake of simplicity. However, this case can be dealt with by following the ideas reported in (Liu & Ozay, 2014).

In the sequel we denote by the pair $(C(L), \mathcal{R}^0_{C(L)})$ the solution to Problem 1, where we emphasize its dependence on the specification L. For later purposes, we also denote the controller solving Problem 1 with specification L by

$$C(L) = (X_{C(L)}, X_{C(L),0}, \mathbf{U}, f_{C(L)}, X_{C(L),m}, Y_{C(L)}, H_{C(L)}). \tag{4.5}$$

Following the approach based on discrete abstractions (also called symbolic systems), the methodology we use to solve Problem 1 consists of three steps:

1) construct a symbolic system that approximates the plant P;

2) solve the control problem at the symbolic level;

3) derive from the previous step the solution to the original control problem.

4.3.1 Symbolic models for nonlinear systems

Results reported in this section are adapted from (Pola & Di Benedetto, 2019; Tabuada, 2009).

Given the nonlinear control system P in (4.1), define the following system representation of P:

$$S(P) = (X, X_0, U, f, X_m, \Gamma, Y, H), \tag{4.6}$$

where:

- $X = \mathbf{X}$;
- $X_0 = \mathbf{X}_0$;
- $U = \mathcal{U}$;
- $x' = f(x, u)$ if $x' = \mathbf{x}(t, x, u)$ for some $t \in \mathbb{R}^+$;
- $X_m = \mathbf{X}$;
- $\Gamma(x) = \mathcal{U}$ for all $x \in X$, namely all inputs are enabled for all states.
- $Y = \mathbf{X}$ and
- $H(x) = x$ for all $x \in X$.

System $S(P)$ is metric when we regard $Y \subseteq \mathbb{R}^n$ as being equipped with a metric. In the sequel we use as metric, the infinity norm, i.e.

$$\mathbf{d}(y, y') = \|y - y'\|, \quad \forall y, y' \in Y.$$

There are strong connections between P and $S(P)$:

- P and $S(P)$ have an infinity number of states;

- P admits a unique solution for any initial state $x(0) \in \mathbf{X}_0$ and for any control input function u, and $S(P)$ is deterministic;
- P is forward complete and $S(P)$ is alive;
- $S(P)$ preserves reachability properties of P, that is, a state x_f is reached by a trajectory of P starting from $\mathbf{X}_0$ if and only if there exists a state run in $S(P)$ ending in x_f.

In deriving a symbolic system for P we first proceed with a time discretization and then with a state space quantization of $S(P)$.
We start with the time discretization. Given P and a sampling time $\tau \in \mathbb{R}^+$, define

$$S_\tau(P) = (X_\tau, X_{\tau,0}, U_\tau, f_\tau, X_{\tau,m}, \Gamma_\tau, Y_\tau, H_\tau), \tag{4.7}$$

where:

- $X_\tau = \mathbf{X}$;
- $X_{\tau,0} = \mathbf{X}_0$;
- U_τ is the set of constant input functions $u : [0, \tau[\to \mathbf{U}$;
- $x' = f_\tau(x, u)$ if $x' = \mathbf{x}(\tau, x, u)$ with $u \in U_\tau$;
- $X_{\tau,m} = \mathbf{X}$;
- $\Gamma_\tau(x) = U_\tau$ for all $x \in X_\tau$;
- $Y_\tau = \mathbf{X}$;
- $H_\tau(x) = x$ for all $x \in X_\tau$.

Some related properties of $S(P)$ and $S_\tau(P)$ are:

- $S(P)$ and $S_\tau(P)$ have an infinity number of states;

- $S(P)$ and $S_\tau(P)$ are deterministic;
- $S(P)$ and $S_\tau(P)$ are alive;
- $S_\tau(P)$ is a subsystem of $S(P)$;
- $S(P)$ and $S_\tau(P)$ are metric.

We now introduce a system that is obtained by quantizing the states of $S_\tau(P)$. Given P, a sampling time $\tau \in \mathbb{R}^+$ and a state space quantization $\eta \in \mathbb{R}^+$, define

$$S_{\tau,\eta}(P) = (X_{\tau,\eta}, X_{\tau,\eta,0}, U_{\tau,\eta}, f_{\tau,\eta}, X_{\tau,\eta,m}, \Gamma_{\tau,\eta}, Y_{\tau,\eta}, H_{\tau,\eta}),$$

where:

- $X_{\tau,\eta} = [\mathbf{X}]^n_\eta$;
- $X_{\tau,\eta,0} = [\mathbf{X}_0]^n_\eta$;
- $U_{\tau,\eta} = U_\tau$, the set of constant input functions $u : [0, \tau[\to \mathbf{U}$;
- $\xi' = f_{\tau,\eta}(\xi, u)$ if $\xi' = [\mathbf{x}(\tau, \xi, u)]^n_\eta$ with $u \in U_{\tau,\eta}$;
- $X_{\tau,\eta,m} = [\mathbf{X}]^n_\eta$;
- $\Gamma_{\tau,\eta}(x) = U_{\tau,\eta}$ for all $x \in X_{\tau,\eta}$;
- $Y_{\tau,\eta} = \mathbb{R}^n$ and
- $H_{\tau,\eta}(x) = x$ for all $x \in X_{\tau,\eta}$.

Intuitively, this definition corresponds to replacing any state of $S_\tau(P)$ by its quantization. Relationships between $S_\tau(P)$ and $S_{\tau,\eta}(P)$ are:

- $S_\tau(P)$ has an infinity number of states while $S_{\tau,\eta}(P)$ has a countable number of states;
- $S_\tau(P)$ and $S_{\tau,\eta}(P)$ are deterministic;

- $S_\tau(P)$ and $S_{\tau,\eta}(P)$ are alive;
- $S_{\tau,\eta}(P)$ is not a subsystem of $S_\tau(P)$.

System $S_{\tau,\eta}(P)$ is countable and becomes symbolic when the set $\mathbf{X}$ is bounded. In this section, the results will be derived under some stability assumptions of the plant P that are briefly explained hereafter.

We start by recalling the notion of incremental global asymptotic stability, which corresponds to the case where all trajectories of a nonlinear control system with the same input converge to each other when time goes to infinity.

Definition 31 *(Angeli, 2002) A control system P is incrementally globally asymptotically stable (δ–GAS) if it is forward complete and there exist a $\mathcal{KL}$ function β such that for any $t \in \mathbb{R}_0^+$, any $x, x' \in \mathbb{R}^n$ and any $\mathbf{u} \in \mathcal{U}$ the following condition is satisfied:*

$$\|\mathbf{x}(t, x, \mathbf{u}) - \mathbf{x}(t, x', \mathbf{u})\| \leq \beta(\|x - x'\|, t). \tag{4.8}$$

The notion of stability defined above is with respect to trajectories and can be thought of as an incremental version of the classical notion of global asymptotic stability (GAS), see e.g. (Khalil, 1996). In general, the inequality (4.8) is difficult to check directly. Fortunately δ–GAS can be characterized by Lyapunov function.

Definition 32 *Consider a control system P and a smooth function*

$$V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_0^+.$$

Function V is called a δ–GAS Lyapunov function for P, if there exist $\kappa \in \mathbb{R}^+$ and $\mathcal{K}_\infty$ functions $\underline{\alpha}$ and $\overline{\alpha}$ such that:

(i) for any $x, x' \in \mathbb{R}^n$

$$\underline{\alpha}(\|x - x'\|) \leq V(x, x') \leq \overline{\alpha}(\|x - x'\|);$$

(ii) for any $x, x' \in \mathbb{R}^n$ *and any* $u \in U$

$$\frac{\partial V}{\partial x} F(x, u) + \frac{\partial V}{\partial y} F(x', u) < -\kappa V(x, x').$$

Theorem 2 *(Angeli, 2002) Control system* P *is* δ*–GAS if it admits a* δ*–GAS Lyapunov function.*

We also assume the existence of a $\mathcal{K}_\infty$ function γ such that

$$\forall x, y, z \in \mathbb{R}^n, \ \|V(x, y) - V(x, z)\| \leq \gamma(\|y - z\|). \tag{4.9}$$

Note that γ is not a function of the variable x. This assumption is not restrictive provided that we are interested in the dynamics of P on a compact subset of the state space $\mathbb{R}^n$, as it is the case in concrete applications.

In order to properly relate systems $S_\tau(P)$ and $S_{\tau,\eta}(P)$ we introduce the following

Theorem 3 *Consider the control system* P *and suppose it admits a* δ*–GAS Lyapunov function* V *and, as such, satisfying conditions of Definition 32, for some* $\kappa \in \mathbb{R}^+$ *and* $\mathcal{K}_\infty$ *functions* $\underline{\alpha}$ *and* $\overline{\alpha}$ *and (4.9) for some* $\mathcal{K}_\infty$ *function* γ*. Then, for any desired accuracy* $\mu \in \mathbb{R}^+$ *and any sampling time* $\tau \in \mathbb{R}^+$*, select quantization parameter* $\eta \in \mathbb{R}^+$ *satisfying:*

$$\eta \leq \min\left\{\gamma^{-1}((1 - e^{-\kappa\tau})\alpha_1(\mu)), (\alpha_2^{-1} \circ \alpha_1)(\mu)\right\}. \tag{4.10}$$

Then, relation $\mathcal{R}_\mu \subseteq X_\tau \times X_{\tau,\eta}$, specified by $(x,\xi) \in \mathcal{R}_\mu$ iff $V(x,\xi) \leq \alpha_1(\mu)$, is a strong μ–approximate bisimulation relation between $S_\tau(P)$ and $S_{\tau,\eta}(P)$. Consequently, $S_\tau(P)$ and $S_{\tau,\eta}(P)$ are strong approximately bisimilar with accuracy μ.

4.3.2 Solution to the symbolic control problem

In this section we use the symbolic system $S_{\tau,\eta}(P)$ defined in the previous section to derive a solution to Problem 1.

We first recall by Chapter 2 that given a regular language L as specification we can always define a symbolic system

$$S_L = (X_L, X_{L,0}, U_L, f_L, X_{L,m}, \Gamma_L, Y_L, H_L)$$

such that $\mathcal{L}_m^y(S_L) = L$. In order to find the controller $C(L)$ solving Problem 1 we start selecting transitions from S_L which can be followed by symbolic system $S_{\tau,\eta}(P)$, up to accuracy η. To this purpose, consider function

$$\mathcal{I} : X_L \times X_L \times \mathbb{R}^+ \times \mathbb{R}^+ \to \{\texttt{True}, \texttt{False}\}. \tag{4.11}$$

For any transition $x^+ \in f_L(x)$ of S_L set

$$\mathcal{I}(x, x^+, \tau, \eta) = \texttt{True}, \tag{4.12}$$

if there exists $u \in \mathbf{U}$ such that

$$[H_L(x^+)]_\eta^n \in f_{\tau,\eta}([H_L(x)]_\eta^n, u),$$

and $\mathcal{I}(x, x^+, \tau, \eta) = \texttt{False}$, otherwise. This means that $\mathcal{I}(x, x^+, \tau, \eta)$ is True, if

the transition $x^+ \in f_L(x)$ of S_L can be matched by system $S_{\tau,\eta}(P)$, up to accuracy η, and False, otherwise.

Define the subsystem of S_L,

$$\widehat{S}_L = (\widehat{X}_L, \widehat{X}_{L,0}, \widehat{U}_L, \widehat{f}_L, \widehat{X}_{L,m}, \widehat{Y}_L, \widehat{H}_L), \tag{4.13}$$

where $x^+ \in \widehat{f}_L(x)$ if $x^+ \in f_L(x)$ and (4.12) is matched.

System $\widehat{S}_L$ is blocking in general. For this reason we define

$$Trim(\widehat{S}_L) = (X_T, X_{T,0}, U_T, f_T, X_{T,m}, Y_T, H_T) \tag{4.14}$$

which is, by definition of $Trim$, accessible and coaccessible and hence nonblocking. We derive the solution to our control problem under the following

Assumption 2 *System $Trim(\widehat{S}_L)$ is not empty.*

Specify entities for controller $C(L)$ in (4.5) as

$$\begin{aligned} &X_{C(L)} = X_T, \\ &X_{C(L),0} = X_{T,0}, \\ &f_{C(L)}(x) = f_T(x) \text{ for all } x \in X_T, \\ &H_{C(L)}(x) = u, \end{aligned} \tag{4.15}$$

where u is such that

$$[H_L(x^+)]^n_\eta = f_{\tau,\eta}([H_L(x)]^n_\eta, u),$$

for some $x^+ \in f_T(x)$ (the existence of x^+ is guaranteed by Assumption 2).

Define the following relation

$$\mathcal{R}^0 = \{(x_0, x_{c,0}) \in \mathbf{X}_0 \times X_{C(L),0} \mid (x_0, [H_{C(L)}(x_{c,0})]_\eta^n) \in \mathcal{R}_\mu\} \tag{4.16}$$

The following result shows that the illustrated procedure to construct the set of initial states and the controller is formally correct and solves the control problem.

Theorem 4 *Consider control system P and suppose it admits a δ–GAS Lyapunov function V and as such satisfying conditions of Definition 32, for some $\kappa \in \mathbb{R}^+$ and $\mathcal{K}_\infty$ functions $\underline{\alpha}$ and $\overline{\alpha}$ and the inequality (4.9) for some $\mathcal{K}_\infty$ function γ. For any desired accuracy $\theta \in \mathbb{R}^+$ and sampling time $\tau \in \mathbb{R}^+$ select $\mu \in \mathbb{R}^+$ and $\eta \in \mathbb{R}^+$ satisfying (4.10) and $\mu + \eta/2 \leq \theta$. Suppose that Assumption 2 holds. Then, controller C in (4.5) specified by (4.15) and relation $\mathcal{R}^0$ in (4.16) solve Problem 1.*

The proof of the result above can be found in (Pola, Pepe, & Di Benedetto, 2018) for discrete–time nonlinear systems.

Remark 2 *Assumption 2 is not limiting in the sense that if $Trim(\widehat{S}_L) = \varnothing$, then, the notion of approximate bisimulation guarantees the so-called "completeness property" in the control, in an approximating sense, which means that a solution to the control problem exists if and only if it can be found by using the approach proposed here, within some accuracy.*

It is easy to see that there can be the case that $\mathcal{L}_m^y(C_1(L)) = \mathcal{L}_m^y(C_2(L))$ for controllers $C_1(L)$ and $C_2(L)$ with $C_1(L) \neq C_2(L)$, meaning that while controllers are different, they enforce the same part of the specification L on the plant P. When $\mathcal{L}_m^y(C_1(L)) = \mathcal{L}_m^y(C_2(L))$ in the sequel, we write $C_1(L) \approx_L C_2(L)$.

4.4 Control problem formulation

We can now introduce the control problem we consider in this chapter. Consider a regular language $L_{nom} \subseteq \mathcal{Y}^*$, representing the nominal specification that we want to enforce on the plant P. Let $C(L_{nom})$ be the controller that solves Problem 1 with $L = L_{nom}$ and suppose that it has been computed off–line before the controlled system $P^{C(L_{nom})}$ is initialized. Suppose at some time $\mathbf{t} = \mathbf{k}\tau$, $\mathbf{k} \in \mathbb{N}_0$, word $\mathbf{w} : w_0 w_1 ... w_{\mathbf{k}} \in \overline{L_{nom}}$ (where we recall $\overline{L_{nom}}$ is the prefix closure of regular language L_{nom}) has been enforced by the controller $C(L_{nom})$ on the plant P, meaning that $\|x(k\tau) - w_k\| \leq \theta, \forall k \in [0; \mathbf{k}]$, where x is the state trajectory of the plant in $P^{C(L_{nom})}$. Suppose that at time $\mathbf{k}\tau$ some words $\mathbf{w}\,\mathbf{q}_- \in L_{nom}$ become not admissible and that some other words $\mathbf{w}\,\mathbf{q}_+ \notin L_{nom}$ become admissible. The resulting new specification at time $\mathbf{k}\tau$ can then be formalized as a new regular language specification L_{new}, as follows:

$$L_{new} = (w_{\mathbf{k}}(L_{nom}/\mathbf{w}) \setminus L_-) \cup L_+, \tag{4.17}$$

where (we recall that $L_{nom}/\mathbf{w}$ is the language collecting suffixes of words in L_{nom} after their prefix $\mathbf{w}$, and):

- regular language $L_- \subset w_{\mathbf{k}}(L_{nom}/\mathbf{w})$ collects all words $w_{\mathbf{k}}\mathbf{q}_-$ that become unfeasible at time $\mathbf{k}\tau$;
- regular language $L_+ \subset (w_{\mathbf{k}}\mathcal{Y}^*) \setminus (w_{\mathbf{k}}(L_{nom}/\mathbf{w}))$ collects all words $w_{\mathbf{k}}\mathbf{q}_+$ that become feasible at time $\mathbf{k}\tau$.

From the definitions above, it is clear that $L_- \cap L_+ = \varnothing$, from which

$$(w_{\mathbf{k}}(L_{nom}/\mathbf{w}) \setminus L_-) \cup L_+ = (w_{\mathbf{k}}(L_{nom}/\mathbf{w}) \cup L_+) \setminus L_-.$$

In order to deal with the change of the specification, one could solve Problem 1 with $L = L_{new}$. However, this approach is in general not efficient from the computational complexity point of view because in many practical applications, the change of the specification exhibits small modifications with respect to the nominal specification L_{nom} for which the controller $C(L_{nom})$ is supposed to be already available. For this reason in this chapter we consider the following control problem:

Problem 2 *Given controller $C(L_{nom})$, the regular languages L_- and L_+, and the state $x_{c,\mathbf{k}}$ reached by the controller $C(L_{nom})$ in $P^{C(L_{nom})}$ at step $\mathbf{k}$, find controller C_{new} that enforces specification L_{new} on the plant P, on the only basis of knowledge of $C(L_{nom})$, $C(L_-)$ and $C(L_+)$.*

We point out that in the problem above, only controllers $C(L_-)$ and $C(L_+)$ need to be computed at time $t = \mathbf{k}\tau$. Provided that sizes of L_- and L_+ are small enough, such computations can be performed at run–time. This is important because, in this way, controller can reconfigure in response to the specification change that is unpredictable and hence, cannot be modeled before it happens.

4.5 Symbolic control with dynamic regular language specifications

In this section we provide the solution to Problem 2.

4.5.1 Preliminary results

In this subsection we introduce two results concerning systems marking the union of regular languages and systems marking the set difference of regular languages.

These results will be applied in the next subsection to provide the solution to Problem 2. We start with the following definition of union of systems:

Definition 33 *Consider a pair of systems S_1 and S_2 as in Definition 1 where $X_1 \cap X_2 = \varnothing$. The union of S_1 and S_2, denoted $S_1 \sqcup S_2$, is specified by system S as in (1), where:*

- $X = X_1 \cup X_2$;
- $X_0 = X_{0,1} \cup X_{0,2}$;
- $U = U_1 \cup U_2$;
- $f(x,u) := \begin{cases} f_1(x,u) & \text{if } x \in X_1 \wedge u \in \Gamma_1(x) \\ f_2(x,u) & \text{if } x \in X_2 \wedge u \in \Gamma_2(x) \\ undefined & otherwise \end{cases}$;
- $X_m = X_{m,1} \cup X_{m,2}$;
- $\Gamma(x) := \Gamma_1(x) \cup \Gamma_2(x)$;
- $Y = Y_1 \cup Y_2$;
- $H(x) := \begin{cases} H_1(x) & \text{if } x \in X_1 \\ H_2(x) & \text{if } x \in X_2 \end{cases}$.

The following result holds:

Proposition 1 $\mathcal{L}_m^y(S_1 \sqcup S_2) = \mathcal{L}_m^y(S_1) \cup \mathcal{L}_m^y(S_2)$.

Proof: Since $X_1 \cap X_2 = \varnothing$, then $S_1 \sqcup S_2$ evolves either through transitions of S_1 or through transitions of S_2, from which the statement holds. ■

We now address systems difference. Techniques for defining systems marking difference of regular languages are well known for example for discrete–event systems (DES). The *traditional construction* consists in first deriving a pair of

DES S_1 and S_2 marking regular languages L_1 and L_2, respectively. Then a DES S_2^{compl} marking the complement of L_2 is derived. Finally, the product composition $S_1 \times S_2^{compl}$ between S_1 and S_2^{compl} is performed. By construction, indeed the language marked by $S_1 \times S_2^{compl}$ coincides with $L_1 \setminus L_2$. We refer to e.g. (Cassandras & Lafortune, 1999) for technical details about this construction. Since systems we consider in this chapter are equipped with outputs on the states, the procedure described above needs to be generalized. This extension is easy to do. However, as later on discussed (see Remark 3), this extension encounters some limitations in terms of computational complexity and for this reason we propose hereafter an alternative definition of systems difference. The key idea behind the difference of S_1 by S_2 is to retrace, whenever possible, the behaviour of S_2 on S_1, and then unmark those marked states of S_1 that are reached by a marked language in S_2.

Definition 34 *Consider a pair of systems S_1 and S_2 as in Definition 1. The system difference of S_1 by S_2, denoted $S_1 \setminus_s S_2$, is $Trim(S)$, where S is a system as in Definition 1, with:*

- $X = (X_1 \times \{x_D\}) \cup \widehat{X}$, *where*

$$\widehat{X} = \{(x_1, x_2) \in X_1 \times X_2 \mid H_1(x_1) = H_2(x_2)\}$$

  and x_D is a dummy state;

- $X_0 \subseteq X_{0,1} \times (X_{0,2} \cup \{x_D\})$ *defined as follows:*

  - $\forall x_1 \in X_{0,1}, x_2 \in X_{0,2}$, *if* $H_1(x_1) = H_2(x_2)$ *then* $(x_1, x_2) \in X_0$;

  - $\forall x_1 \in X_{0,1}$, *if there does not exist* $x_2 \in X_{0,2}$ *such that* $H_1(x_1) = H_2(x_2)$ *then* $(x_1, x_D) \in X_0$;

- $f : X \times U_1 \to X$ *defined as follows: for any* $x_1 \in X_1$ *and* $u_1 \in \Gamma_1(x_1)$

- $f((x_1, x_2), u_1) := (f_1(x_1, u_1), f_2(x_2, u_2))$, *for any* $u_2 \in U_2$ *for which* $f(x_2, u_2)$ *is defined,*
- $f((x_1, x_D), u_1) := (f_1(x_1, u_1), x_D)$,
- $f((x_1, x_2), u_1) := (f_1(x_1, u_1), x_D)$, *if there does not exist an input* $u_2 \in U_2$ *such that* $(f_1(x_1, u_1), f_2(x_2, u_2)) \in \widehat{X}$;

- $X_m = (X_{m,1} \times \{x_D\}) \cup (X_{m,1} \times (X_2 \setminus X_{m,2})) \cap \widehat{X}$;
- $\Gamma((x_1, x_2)) = \Gamma_1(x_1)$ *for all* $(x_1, x_2) \in X$;
- $Y = Y_1$;
- $H(x_1, x_2) = H_1(x_1)$, *for any* $(x_1, x_2) \in X$.

Definition above proposes a construction that in fact integrates the intermediate steps used in the *traditional construction* and to the best of our knowledge is new. The definition of the transition relation above is composed of three conditions:

- in the first one we synchronize transitions of the two systems S_1 and S_2 with starting states sharing the same output and with ending states sharing the same output;
- in the second condition we allow any transition of S_1;
- in the third condition we allow a transition starting from (x_1, x_2) even when there is no state successor of x_2 in S_2 sharing the same output of the state $f_1(x_1, u_1)$.

The following result holds:

Proposition 2 *If* S_2 *is output deterministic then* $\mathcal{L}_m^y(S_1 \setminus_s S_2) = \mathcal{L}_m^y(S_1) \setminus \mathcal{L}_m^y(S_2)$.

Proof: Since unary operators Ac and $CoAc$ do not modify marked languages of involved systems we have $\mathcal{L}_m^y(S_1 \diagdown_s S_2) = \mathcal{L}_m^y(S)$. By Definition 34, for any output run r_Y in S, there exists a state run in S_1 with output run r_Y and ending in a marked state $x_{m,1}$. If r_Y is also an output run in S_2 then, since S_2 is output deterministic, there exists only a state run, with corresponding output run r_Y, which ends in a marked state $x_{m,2}$ (case 1) or in a state x_2 that is not marked (case 2). By Definition 34, in case 1, state $(x_{m,1}, x_{m,2})$ is set to be not marked while in case 2, state $(x_{m,1}, x_2)$ is set to be marked. By Definition 34, if r_Y is not an output run in S_2 then we get that state $(x_{m,1}, x_D)$ in S is marked. The case where state $x_{m,1}$ is not marked is not relevant because here we are considering only marked languages. ■

For sake of clarity we propose the following example.

Example 10 *Consider system S_1 as in Figure 4.1 and system S_2 as in Figure 4.2.*

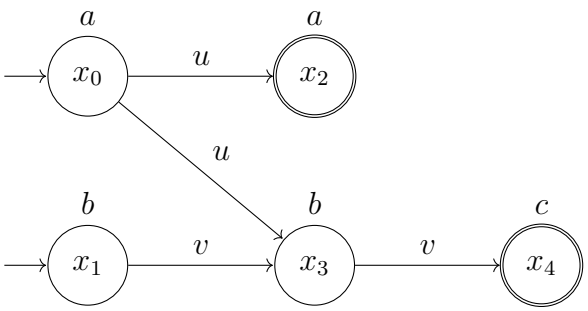

Figure 4.1: System S_1 of Example 10

We construct the system difference of S_1 by S_2 through the following steps.

> *First step, we build the set $X_0 := \{(x_0, x_0'), (x_1, x_D)\}$, where x_0 is paired with x_0' because they have the same output and x_1 is paired with the dummy state x_D because does not exists an initial state of S_2 with output b. At this moment we set $X := X_0$*

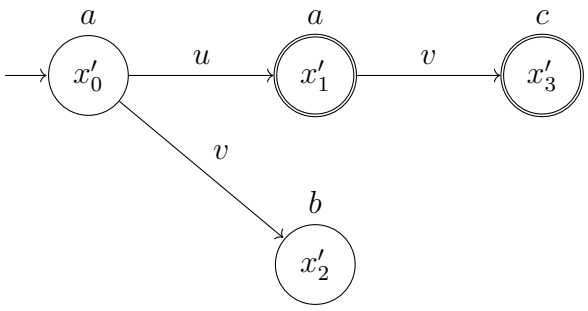

Figure 4.2: System S_2 of Example 10

Next, for each state in X we define the transitions of f and add, in X, all new reached states.

- *So, consider state $(x_0, x'_0) \in X$, from x_0 we may have $x_2 = f_1(x_0, u)$ and from x'_0 the transition $x'_1 = f_2(x'_0, u)$ such that $H_1(x_2) = H_2(x'_1)$. Hence we set $(x_2, x'_1) = f((x_0, x'_0), u)$.*
- *Same considerations for $(x_3, x'_2) = f((x_0, x'_0), u)$, where we point out that does not matter the input of system S_2.*
- *Instead, from state $(x_1, x_D) \in X$, with the second state dummy, the evolution will always be with dummy state, i.e. $(x_3, x_D) = f((x_1, x_D), v)$ and $(x_4, x_D) = f((x_3, x_D), v)$. We do not need to do anything.*

Now we compute the transitions of f from states (x_2, x'_1) and (x_3, x'_2).

- *It is easy to see that, since from x_2 system S_1 does not evolve, for the state (x_2, x'_1) the function f is not defined.*
- *Instead, from x'_2 system S_2 does not evolve, hence we set $(x_4, x_D) = f((x_3, x'_2), v)$*

Last but not least, we define the set of marked states X_m as all pairs that contain a state in $X_{m,1}$ and do not contain a state in $X_{m,2}$, i.e. $X_m := \{(x_4, x_D)\}$. The resulting state difference is depicted in Figure 4.3.

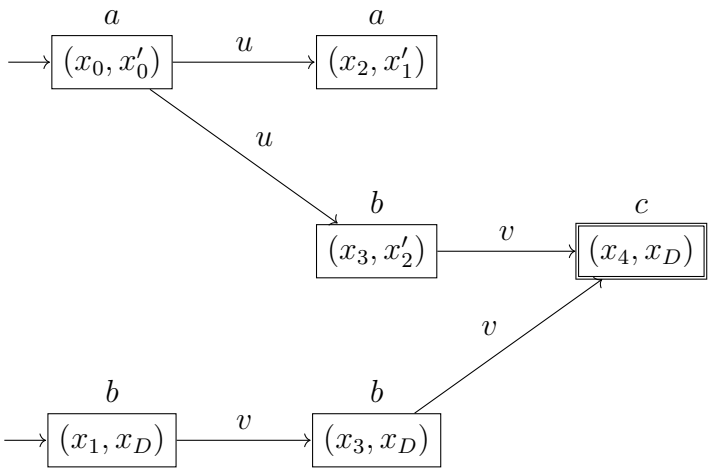

Figure 4.3: System $S = S_1 \diagdown_s S_2$ of Example 10

It is easy to verify Proposition 2, in fact $\mathcal{L}_m^y(S_1) = \{a\,a, a\,b\,c, b\,b\,c\}$, $\mathcal{L}_m^y(S_2) = \{a\,a, a\,a\,c\}$ *and* $\mathcal{L}_m^y(S_1 \diagdown_s S_2) = \{a\,b\,c, b\,b\,c\} = \mathcal{L}_m^y(S_1) \diagdown \mathcal{L}_m^y(S_2)$.

4.5.2 Solution to Problem 2

In this subsection we provide the solution to Problem 2. Without loss of generality we make the following assumption.

Assumption 3 *The sets of states* $X_{C(L_{nom})}$, $X_{C(L_+)}$, *of controllers* $C(L_{nom})$ *and* $C(L_+)$, *respectively, have empty intersection.*

Define regular language

$$L_{next} = w_{\mathbf{k}}(L_{nom}/\mathbf{w}), \tag{4.18}$$

where we recall that $\mathbf{w} = w_0\, w_1 \ldots w_{\mathbf{k}}$.

Definition 35 *Let* $C_{next} = (X_{C_{next}}, X_{C_{next},0}, \mathbf{U}, f_{C_{next}}, X_{C_{next},m}, Y_{C_{next}}, H_{C_{next}})$ *be the controller coinciding with* $C(L_{nom})$ *except for the set of initial states, that for* C_{next} *is defined as* $\{x_{c,\mathbf{k}}\}$, *where we recall (from Problem 2) that* $x_{c,\mathbf{k}}$ *is the state reached by the controller* $C(L_{nom})$ *in* $P^{C(L_{nom})}$ *at step* $\mathbf{k}$.

Intuitively, C_{next} is the controller enforcing all words that can be enforced by $C(L_{nom})$ with prefix $w_0\, w_1 \ldots w_{\mathbf{k}-1}$. It is worth to point out that C_{next} does not coincide in general, with $C(L_{next})$ which is the controller solving Problem 1 with specification L_{next}. However, the following result holds.

Proposition 3 $C_{next} \approx_{L_{next}} C(L_{next})$

Set

$$\widetilde{L} = L_{next} \cup L_{+}. \tag{4.19}$$

Without loss of generality we make the following

Assumption 4 *The sets of states $X_{C(\widetilde{L} \setminus L_{-})}$, $X_{C(L_{-} \setminus \widetilde{L})}$ and $X_{C(\widetilde{L} \cap L_{-})}$ of controllers $C(\widetilde{L} \setminus L_{-})$, $C(L_{-} \setminus \widetilde{L})$ and $C(\widetilde{L} \cap L_{-})$, respectively, have empty intersection.*

We now have all the ingredients to give the main result of this chapter which provides the solution to Problem 2.

Theorem 5 *Controller $C(L_{new})$ solving Problem 2 is such that $C(L_{new}) \approx_{L_{new}} C_{new}$, where:*

$$C_{new} = (C_{next} \sqcup C(L_{+})) \setminus_{s} C(L_{-}).$$

Proof: By Proposition 2, we get:

$$\begin{aligned} &\mathcal{L}_m^y((C_{next} \sqcup C(L_{+})) \setminus_s C(L_{-})) = \\ &\mathcal{L}_m^y(C_{next} \sqcup C(L_{+})) \setminus \mathcal{L}_m^y(C(L_{-})). \end{aligned} \tag{4.20}$$

Since set of states $X_{C_{next}}$ of C_{next} coincides with set of states of $C(L_{nom})$, by

Assumption 3, we get $X_{C_{next}} \cap X_{C(L_+)} = \varnothing$. Hence, by Proposition 1 we get:

$$\begin{aligned}&\mathcal{L}_m^y(C_{next} \sqcup C(L_+)) \setminus \mathcal{L}_m^y(C(L_-)) = \\ &(\mathcal{L}_m^y(C_{next}) \cup \mathcal{L}_m^y(C(L_+))) \setminus \mathcal{L}_m^y(C(L_-)).\end{aligned} \tag{4.21}$$

By Proposition 3, we get:

$$\begin{aligned}&\mathcal{L}_m^y(C_{next}) \cup \mathcal{L}_m^y(C(L_+)) \setminus \mathcal{L}_m^y(C(L_-)) = \\ &\mathcal{L}_m^y(C(L_{next})) \cup \mathcal{L}_m^y(C(L_+)) \setminus \mathcal{L}_m^y(C(L_-)).\end{aligned} \tag{4.22}$$

By Proposition 1 and set theory, and by recalling (4.19), we get:

$$\begin{aligned}&\mathcal{L}_m^y(C(L_{next})) \cup \mathcal{L}_m^y(C(L_+)) \setminus \mathcal{L}_m^y(C(L_-)) = \\ &\mathcal{L}_m^y(C(\widetilde{L})) \setminus \mathcal{L}_m^y(C(L_-)) = \\ &\mathcal{L}_m^y(C((\widetilde{L} \setminus L_-) \cup (\widetilde{L} \cap L_-))) \setminus \\ &\mathcal{L}_m^y(C((L_- \setminus \widetilde{L}) \cup (\widetilde{L} \cap L_-))).\end{aligned} \tag{4.23}$$

By definition of L_- and by (4.18), we get $L_- \subset L_{next}$ which, combined with (4.19), implies $L_- \subset \widetilde{L}$. Hence, $L_- \setminus \widetilde{L} = \varnothing$, which, by Assumption 4 and Proposition 1, implies:

$$\begin{aligned}&\mathcal{L}_m^y(C((\widetilde{L} \setminus L_-) \cup (\widetilde{L} \cap L_-))) \setminus \mathcal{L}_m^y(C(\widetilde{L} \cap L_-)) = \\ &\mathcal{L}_m^y(C(\widetilde{L} \setminus L_-) \sqcup C(\widetilde{L} \cap L_-)) \setminus \mathcal{L}_m^y(C(\widetilde{L} \cap L_-)) = \\ &(\mathcal{L}_m^y(C(\widetilde{L} \setminus L_-)) \cup \mathcal{L}_m^y(C(\widetilde{L} \cap L_-))) \setminus \\ &\mathcal{L}_m^y(C(\widetilde{L} \cap L_-)) = \mathcal{L}_m^y(C(\widetilde{L} \setminus L_-)).\end{aligned} \tag{4.24}$$

Now by combining (4.17), (4.18), (4.19) and recalling that $L_- \cap L_+ = \varnothing$ we get:

$$\mathcal{L}_m^y(C(\widetilde{L} \setminus L_-)) = \mathcal{L}_m^y(C(L_{new})). \tag{4.25}$$

By combining (4.20), (4.21), (4.22), (4.23), (4.24) and (4.25), the statement holds.

■

4.6 Computational complexity analysis

In this section we provide an analysis of space computational complexity (S.c.c.) and time computational complexity (T.c.c.) of the approach we proposed in this chapter and compare it with traditional approaches. We recall that the S.c.c. of a symbolic system S as in Definition 1 is evaluated as $card(f)$. We start with the following:

Proposition 4 *S.c.c. and T.c.c. for finding $C(L)$ of Problem 1 are:*

- *S.c.c. $card(f_L) + card(f_{\tau,\eta})$ and*
- *T.c.c. $card(f_L)\, card(f_{\tau,\eta})$,*

where f_L is the transition relation of the symbolic system S_L with output language $\mathcal{L}_m^y(S_L) = L$.

It is easy to see that analysis of computational complexity of Problem 2 needs to consider S.c.c. and T.c.c. of the following steps:

a) computation of $C(L_-)$;

b) computation of $C(L_+)$;

c) computation of $C_{next} \sqcup C(L_+)$;

d) computation of $C_{new} = (C_{next} \sqcup C(L_+)) \diagdown_s C(L_-)$.

S.c.c. and T.c.c. of steps a) and b) can be evaluated by Proposition 4. The following result considers step c).

Proposition 5 *S.c.c. and T.c.c. to compute* $C_{next} \sqcup C(L_+)$ *are* $card(f_{C_{next}}) + card(f_{C(L_+)})$ *and constant, respectively.*

We point out that T.c.c. in computing $C_{next} \sqcup C(L_+)$ is constant because all entities of operator $\sqcup$ are explicitly defined in Definition 33 and hence no computation is needed. We now consider step d); we introduce notation $C_{n+} = C_{next} \sqcup C(L_+)$. The following holds:

Proposition 6 *S.c.c. and T.c.c. to compute* $C_{n+} \diagdown_s C(L_-)$ *are* $2\, card(f_{C_{n+}})$ *and* $card(f_{C_{n+}})\, card(f_{C(L_-)})$*, respectively.*

Proof: By Definition 34, transitions of $C_{n+} \diagdown_s C(L_-)$ are transitions of C_{n+} either matching transitions of $C(L_-)$ or not or with dummy state x_D. Hence, it is readily seen that resulting S.c.c. is $2\, card(f_{C_{n+}})$. T.c.c. follows from the fact that we need to compare each transition of C_{n+} with each transition of $C(L_-)$. ■

We can now give the main result of this section.

Theorem 6 *S.c.c. for solving Problem 2 is:*

$$2\, card(f_{C_{next}}) + card(f_{L_-}) + card(f_{L_+}) + card(f_{\tau,\eta}). \tag{4.26}$$

T.c.c. for solving Problem 2 is:

$$\begin{aligned} &(card(f_{L_-}) + card(f_{L_+}))\, card(f_{\tau,\eta}) + \\ &(card(f_{C_{next}}) + card(f_{C(L_+)}))\, card(f_{C(L_-)}). \end{aligned} \tag{4.27}$$

Proof: The result holds as a direct application of Propositions 4, 5 and 6. ■

A traditional approach to solve Problem 2 consists in solving Problem 1 with

$L = L_{new}$ whose S.c.c. and T.c.c., by Proposition 4, are, respectively

$$card(f_{L_{new}}) + card(f_{\tau,\eta}), \tag{4.28}$$

$$card(f_{L_{new}})\, card(f_{\tau,\eta}). \tag{4.29}$$

A comparison between the two approaches follows. We suppose that

$$card(f_{L_-}), card(f_{L_+}) << card(f_{\tau,\eta}), \tag{4.30}$$

$$card(f_{L_-}), card(f_{L_+}) << card(f_{L_{new}}). \tag{4.31}$$

Condition above corresponds to the case where the specification change is very small with respect to the nominal situation, which is the case in many realistic scenarios. Since (4.31) implies $card(f_{L_-}), card(f_{L_+}) << card(f_{L_{nom}})$ and $card(f_{L_{nom}}) \sim card(f_{L_{new}}) \sim card(f_{C(L_{new})}) \sim card(f_{C_{next}})$, by comparing (4.26) and (4.28), we obtain that S.c.c. in the two approaches is of the same order of computational complexity. Regarding T.c.c. we get

Proposition 7 *T.c.c. in (4.27) is smaller than T.c.c. in (4.29), under inequalities (4.30) and (4.31).*

Proof: By (4.31), we get $card(f_{L_-})\, card(f_{\tau,\eta})$, $card(f_{L_+})\, card(f_{\tau,\eta}) << card(f_{L_{new}})\, card(f_{\tau,\eta})$. By (4.30), by recalling that sizes of C_{next} and $C(L_{new})$ are comparable, and that a controller system is smaller than the specification system that enforces, we get $card(f_{C_{next}})\, card(f_{C(L_-)}) << card(f_{L_{new}})\, card(f_{\tau,\eta})$. Hence, $card(f_{C(L_-)})\, card(f_{C(L_+)}) << card(f_{L_{new}})\ card(f_{\tau,\eta})$. As a consequence, the statement holds. ■

We conclude this section with the following

Remark 3 *Here we give S.c.c. and T.c.c. related to our technique to compute systems marking difference of regular languages (Definition 34), and the traditional construction recalled after Proposition 1. By using standard techniques to compute* S_2^{compl}*, see e.g. (Cassandras & Lafortune, 1999), we get* $card(f_{2,\,compl}) \sim (card(f_2)+card(Y_1)^2)$*. Hence S.c.c. and T.c.c. for constructing* $S_1 \times S_2^{compl}$ *are given by*

$$\begin{aligned} &\textit{S.c.c.: } card(f_1) + card(f_2) + card(Y_1)^2, \\ &\textit{T.c.c.: } card(f_1)\,(card(f_2) + card(Y_1)^2). \end{aligned}$$

For $S_1 = C(L_{nom})$ *and* $S_2 = C(L_-)$ *we get*

$$\begin{aligned} &\textit{S.c.c.: } card(f_{C(L_{nom})}) + card(f_{C(L_-)}) + card(Y_{C(L_{nom})})^2, \\ &\textit{T.c.c.: } card(f_{C(L_{nom})})\,(card(f_{C(L_-)}) + card(Y_{C(L_{nom})})^2). \end{aligned}$$

which, compared with results in Proposition 6, shows that under (4.31) our approach performs better.

4.7 Illustrative example

Consider plant

$$P : \begin{cases} \dot{x_1} = -4x_1 + 2x_2, \\ \dot{x_2} = -4x_2 + \sin(x_2) + cos(x_1) + u, \\ (x_1, x_2) \in \mathbf{X} = [-0.5, 0.5] \times [-1, 1], \\ \mathbf{X}_0 = \mathbf{X}, \\ u \in \mathbf{U} = 0.3\,\mathbb{Z} \cap [-3, 3]. \end{cases} \tag{4.32}$$

We consider a reach–avoid specification, consisting in reaching in finite time, from

a set of initial states $\mathbb{I} \subset \mathbf{X}$, a target set $\mathbb{T} \subset \mathbf{X}$, while avoiding an obstacle set $\mathbb{O} \subset \mathbf{X}$, where:

- the initial states are $\mathbb{I} = 2\eta\,([2;3] \times [-5;-7])$;
- the target states are $\mathbb{T} = 2\eta\,([1;3] \times [-1;1] \cup [2;3] \times [6;7])$;
- the states to avoid are $\mathbb{O} = 2\eta([-3;0] \times [4;7] \cup [-1;0] \times [-4;0] \cup [1;3] \times [-4;-3])$.

We recall that the reach–avoid specification above can be represented as a regular language, see e.g. Example 4.6 of (Pola & Di Benedetto, 2019). In the sequel we denote this regular language by L_{nom}.

By following the results recalled in Section 6.3, we first checked that indeed P is incrementally asymptotically stable. To this purpose, consider function $V : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_0^+$ defined by $V(x, x') = \|x - x'\|$, for all $x, x' \in \mathbb{R}^2$. It is easy to see that V satisfies conditions of Definition 32 with $\kappa = 2$ and $\alpha_1(\|x - x'\|) = \alpha_2(\|x - x'\|) = \|x - x'\|$. Moreover, it is possible to show that the inequality (4.9) is satisfied with $\mathcal{K}_\infty$ function specified by $\gamma(\|y - z\|) = \|y - z\|$. By applying Theorem 3, for a desired accuracy $\theta = 0.2$, a desired clock period $\tau = 0.2$, and for obtained $\kappa = 2$ we can pick a quantization parameter $\eta = 0.0659$. The obtained symbolic system $S_{\tau,\eta}(P)$ consists of 105 states and 1136 transitions, and it is depicted in Figure 4.4.

The controller $C(L_{nom})$ enforcing specification L_{nom} consists of 45 states and 154 transitions; the time of computation is 3.2 s and it is shown in Figure 4.5.

We suppose that at time $\mathbf{k}\tau$, with $\mathbf{k} = 6$, the specification changes and this is formalized by the regular languages

- $L_{\_} = \{(0, 4\eta)\,(2\eta, 4\eta)\,(2\eta, 0)\}$ and

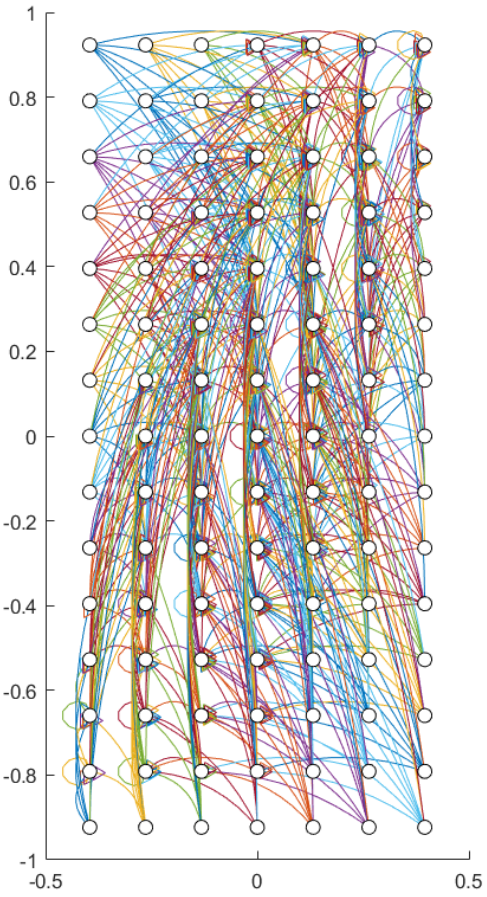

Figure 4.4: Symbolic system $S_{\tau,\eta}(P)$.

- $L_+ = \{(0, 4\eta)\,(0, 0)\,,(0, 4\eta)\,(2\eta, 4\eta)\,(2\eta, 2\eta)\,(0, -2\eta)\}$.

Word enforced by the controller $C(L_{nom})$ on the plant P from $t = 0$ to $t = \mathbf{k}\tau$ is

$$\mathbf{w} = (4\eta, -14\eta)\,(-2\eta, -12\eta)\,(-4\eta, -12\eta)\,(-4\eta, -8\eta)\,(-4\eta, -2\eta)\,(-2\eta, 4\eta)\,(0, 4\eta).$$

By using the results illustrated in Section 6.4, we computed controller C_{new}, which solves Problem 2. This controller is depicted in Figure 4.6, where initial states are $\{1, 28\}$, marked states are $\{2, 3, 5, 12, 13, 14, 19, 20, 24, 25, 26, 27\}$ and output function is detailed in Table 4.1. Note that the initial states 1 and 28 have the same

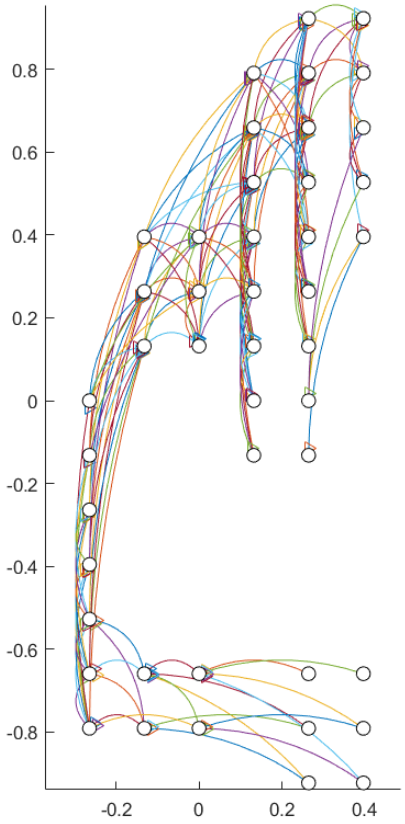

Figure 4.5: Controller $C(L_{nom})$.

output $(0, 4\eta)$ as the one of state $x_{c,\mathbf{k}}$ of $C(L_{nom})$, in accordance with the results presented in the previous sections, i.e. Definition 33. Such a controller consists of 30 states and 90 transitions and its time of computation is 0.17 s.

In order to compare our approach with the traditional one, we also computed controller $C(L_{new})$, by following the results recalled in Section 6.3, and obtained a controller consisting of 27 states and 85 transitions; its time of computation is 1.9 s. S.c.c. in computing C_{new} and $C(L_{new})$ are then comparable, whereas T.c.c. in computing C_{new} is less than 10 times T.c.c. in computing $C(L_{new})$. Note that since $\tau = 0.2$ s, the design of our controller C_{new} can be done at run time, instead the design of the traditional controller $C(L_{new})$ cannot. Figure 4.7 depicts

State	Output
1	$(0, 4\eta)$
2	$(2\eta, -2\eta)$
3	$(2\eta, 0)$
4	$(2\eta, 0)$
5	$(2\eta, 2\eta)$
6	$(2\eta, 4\eta)$
7	$(2\eta, 4\eta)$
8	$(2\eta, 6\eta)$
9	$(2\eta, 8\eta)$
10	$(2\eta, 10\eta)$
11	$(2\eta, 12\eta)$
12	$(4\eta, -2\eta)$
13	$(4\eta, 0)$
14	$(4\eta, 2\eta)$
15	$(4\eta, 4\eta)$

State	Output
16	$(4\eta, 6\eta)$
17	$(4\eta, 8\eta)$
18	$(4\eta, 10\eta)$
19	$(4\eta, 12\eta)$
20	$(4\eta, 14\eta)$
21	$(6\eta, 6\eta)$
22	$(6\eta, 8\eta)$
23	$(6\eta, 10\eta)$
24	$(6\eta, 12\eta)$
25	$(6\eta, 14\eta)$
26	$(0, -2\eta)$
27	$(0, 0)$
28	$(0, 4\eta)$
29	$(2\eta, 2\eta)$
30	$(2\eta, 4\eta)$

Table 4.1

one evolution of the state variables of the controlled plant. More precisely, red line depicts the state evolution $x : [0, 9\tau] \rightarrow \mathbf{X}$ of P, blue dots are states $x(k\tau)$ with $k \in [0; 9]$. Circles and arrows connecting them, represent a word of the specification $\mathbf{w}L_{new}$ enforced on the plant. Since accuracy is $\theta = 0.2$, it is easy to see that the specification is met.

Computations have been performed on a Matlab suite, by using a laptop with CPU Intel Core i7-6700HQ at 2.60 GHz.

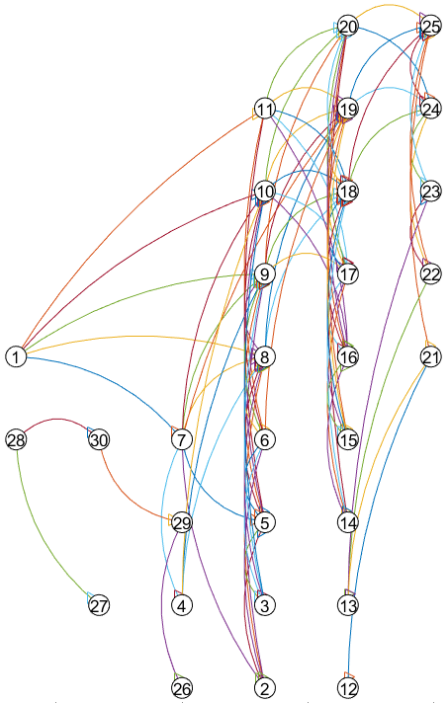

Figure 4.6: Controller C_{new}.

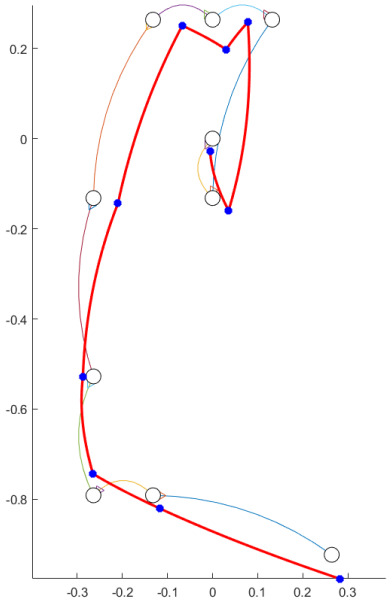

Figure 4.7: One evolution of the state variables of the controlled plant and corresponding word of the specification enforced.

5 | Output feedback control of finite systems with reach avoid specifications

This chapter addresses control design of nondeterministic finite state systems with reachability specifications and reach-avoid specifications. The class of controllers we use is rather general and combines feedforward and output feedback schemes. The proposed controller allows not only reachability (and safe reachability) of the desired target set but also the identification of the reached state in the target set. The solution to the investigated problem has important implications in the context of recovery control and symbolic control design of nonlinear and hybrid systems, as discussed also through some examples. This chapter is mainly based on the work (Masciulli, Pola, De Santis, & Di Benedetto, 2021).

5.1 Introduction

Obtaining a safe architecture coupled with reachability verification and control design for discrete, continuous and hybrid systems have been extensively examined in the computer science, systems and control engineering communities, (see e.g. (Cassandras & Lafortune, 1999; Clarke, Grumberg, & Peled, 1999; Bogomolov & Jungers, 2021)).

Apart from the interest *per se* from a theoretical viewpoint, reachability plays an important role in the analysis and synthesis problems of many natural or artificial processes of interest, such as biological systems, air traffic management systems, robotics, chemical processes, etc.

The proposed solution controller combines feedforward and output feedback schemes, thus allowing not only reachability of the desired target set (as requested by the reachability specification) but also the identification of the state that has

been reached in the target set. This result has important implications in the context of *recovery control* of discrete and continuous systems and also *symbolic control design* of nonlinear and hybrid systems, as discussed also through some examples.

In *recovery control*, the idea is that starting from a nominal working operation of a system, faults and/or (cyber) attacks may deteriorate the performances that should be guaranteed. It is then important to design a control strategy that recovers the nominal situation from the information available on the system. This problem can be handled by using the general results presented in this chapter.

In *symbolic control design*, the goal is to design a digital controller that enforces logic type specifications on purely continuous or hybrid systems. This approach is relevant in many domains including cyber-physical systems and systems biology. The main idea in this approach is to approximate the process to be controlled through a finite state system in order to design a controller on the approximated system, see e.g. (Pola & Di Benedetto, 2019; Tabuada, 2009). Most of the existing results however, are based on full knowledge of the state of the process to control. The results we propose in this chapter provide a method for addressing symbolic control design only based on systems' outputs.

Works that have connections with our results are (Guo, 2018; Zhang, Xia, & Chen, 2020; Wu, Yan, Lin, & Lan, 2009). A brief discussion follows. Work (Guo, 2018) proposes several notions of observability for the class of Boolean Control Networks (BCN). Among those notions, it proposes and characterizes output-feedback observability which deals with designing a dynamic output-feedback controller ensuring the reconstruction of the initial state of the BCN. The focus in

(Guo, 2018) is on the reconstruction of the initial state, while our work focuses on the reconstruction of a state belonging to a desired target set. In the case of deterministic systems and if the target set coincides with the whole state space, the two control problems coincide. Work (Zhang et al., 2020) deals with stabilization of nondeterministic Moore-type finite systems via static output feedback. Our model is more general than the one in (Zhang et al., 2020) because it considers a set of initial states. More importantly, while the control objective of (Zhang et al., 2020) is to steer the state of the system in a controlled equilibrium point via static output feedback, our control objective is to steer the state of the system in a desired target set via dynamic output feedback, asking also for the reached state in the target set to be identified. In (Wu et al., 2009) controllers for enforcing motion planning specifications on robots are proposed. While in (Wu et al., 2009) the model is linear affine and the controller is static output feedback, our model is a finite state system and our controller is a dynamic output feedback controller; moreover, our specifications are less general than specifications in (Wu et al., 2009).

5.2 Reachability problem formulation

In this section, we introduce some preliminary definitions that are needed to formulate the control problem. Furthermore, we highlight that the class of system considered in this chapter is that of Definition 7. We recall that for these systems $\varepsilon \notin U$ and $\varepsilon \notin Y$.

Consider an evolution $\widehat{x}$ of system S and a partition $\{\widehat{x}_j\}_{j=0,\ldots,k}$ of it. By Definition 9, each $\widehat{x}_j$ is an evolution of S. In the sequel, we abuse notation by writing string $H(\widehat{x})$ instead of string $H(\widehat{x}(1))\, H(\widehat{x}(2)) \ldots$. Inputs and outputs associated with an evolution of S are captured by the notion of behaviour, as follows.

Definition 36 *Let $\widehat{x}$ be an evolution of S. Consider a partition $\{\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_k\}$*

of $\widehat{x}$ *with* $\widehat{x}_0 \in X_0$, $y_0 = H(\widehat{x}_0) \in Y$, *and* $\widehat{x}_j \in X^*$, $\widehat{y}_j = H(\widehat{x}_j) \in Y^*$ *for all* $j = 1, \ldots, k$. *Let* $\widehat{u}_{j+1}$ *be a string of inputs associated to the evolution* $\widehat{x}_j$ *such that* $\widehat{x}_{j+1}(1) \in f(\widehat{x}_j(end), \widehat{u}_{j+1}(end))$ *for all* $j = 0, \ldots, k-1$. *The string* $\beta = y_0\, \widehat{u}_1\, \widehat{y}_1 \ldots \widehat{u}_k\, \widehat{y}_k\, \widehat{u}_{k+1}$ *is a behaviour of* S.

For later purposes, for a behaviour $\beta = y_0\, \widehat{u}_1\, \widehat{y}_1 \ldots \widehat{u}_k\, \widehat{y}_k\, \widehat{u}_{k+1}$, we set $N(\beta) = 2\,k + 2$, corresponding to the cardinality of the set $\{y_0, \widehat{u}_1, \widehat{y}_1, \ldots, \widehat{u}_k, \widehat{y}_k, \widehat{u}_{k+1}\}$. In the sequel it is useful to consider a subset of behaviours of S as follows:

Definition 37 *The set* $\mathcal{B}$ *is the collection of behaviours* $\beta = y_0\, \widehat{u}_1\, \widehat{y}_1 \ldots \widehat{u}_k\, \widehat{y}_k\, \widehat{u}_{k+1}$ *of system* S *such that* $Y' = \{\widehat{y} \in Y_\varepsilon^* \mid y_0\, \widehat{u}_1\, \widehat{y}_1 \ldots \widehat{u}_k\, \widehat{y}\, \widehat{u}_{k+1}$ *is a behaviour of* $S\}$ *is a distinguishable set. Let* $X(\beta)$ *be the set of all ending states of all evolutions associated to a behaviour* $\beta \in \mathcal{B}$ *of* S.

The following example illustrates the definitions introduced above.

Example 11 *Consider system depicted in Fig. 5.1.*

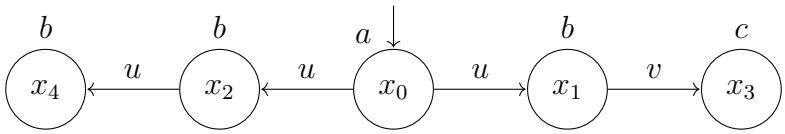

Figure 5.1: System S of Example 11.

By Definition 9, string $x_0\, x_1\, x_3$ *is an evolution of* S; *indeed, a possible input string associated with* x_0 *is* $v\, v\, u$, *with* x_1 *is* $u\, v$ *and with* x_3 *is* ε. *Consider string* $\beta = y_0\, \widehat{u}_1\, \widehat{y}_1\, \widehat{u}_2\, \widehat{y}_2$ *with* $y_0 = a$, $\widehat{u}_1 = v\, v\, u$, $\widehat{y}_1 = b$, $\widehat{u}_2 = u\, v$ *and* $\widehat{y}_2 = c$. *By Definition 36, string* β *is a behaviour associated with evolution* $x_0\, x_1\, x_3$ *with partition* $\{x_0, x_1, x_3\}$. *Note that* β *is not the unique behaviour associated to* $x_0\, x_1\, x_3$; *indeed, by considering partition* $\{x_0, x_1\, x_3\}$, *also* $y_0\, \widehat{u}_1\, \widehat{y}_1$, *with* $y_0 = a$, $\widehat{u}_1 = v\, v\, u\, u\, v$ *and* $\widehat{y}_1 = b\, c$, *is so. We now discuss Definition 37.*

Consider behaviour $\beta = a\,v\,v\,u\,b\,u\,v\,c$ defined above. Resulting set Y' associated with $a\,v\,v\,u\,b\,u\,v$ is $\{b, c\}$. By Definition 13, Y' is a distinguishable set, hence $\beta \in \mathcal{B}$. Conversely, consider strings $a\,u\,u\,b$ and $a\,u\,u\,b\,b$; by Definition 36, they are behaviours of S. By Definition 12, strings b and $b\,b$ are indistinguishable; as a consequence, strings $a\,u\,u\,b$ and $a\,u\,u\,b\,b$ are not in $\mathcal{B}$. We cannot enumerate all behaviours in $\mathcal{B}$ because they are infinity. The unique evolution associated with β is $x_0\,x_1\,x_3$ and as a consequence, $X(\beta) = \{x_3\}$.

We are now ready to present our control problem consisting in designing a controller C based on the output information which brings the state of the system S in a desired set X_m; moreover, the reached marked state has to be identified. More formally:

Problem 3 *Given system S, find conditions for the existence of a controller described by the partial function $C : \mathcal{B} \to U^*$ and $k_{max} \in \mathbb{N}$ such that for any $\beta = y_0\,\widehat{u}_1\,\widehat{y}_1 \cdots \in \mathcal{B}$, with $\widehat{u}_k = C(\beta_{k-1})$ and $\beta_{k-1} = y_0\,\widehat{u}_1\,\widehat{y}_1 \ldots \widehat{u}_{k-1}\,\widehat{y}_{k-1}$, there exists $\mathbf{k} \leq k_{max}$ such that*

$$X(\beta_{\mathbf{k}}) = \{x_m\} \;\wedge\; x_m \in X_m.$$

Control strategies considered in Problem 3 are rather general and mix feedforward and feedback schemes. The following example aims at clarifying the need for such general controller types.

Example 12 *Consider system S in Fig. 5.2 and the behaviour $\beta_1 = a\,u\,b$. Resulting set is $X(\beta_1) = \{x_3, x_4\}$. At this stage one can either choose a control input or a string of control inputs. Since it is not known if current state of S is x_3 or x_4, the choice of control input $C(\beta_1) = u$ does not guarantee to reach marked state x_6; the same reasoning applies for the choice of $C(\beta_1) = v$. Hence, in*

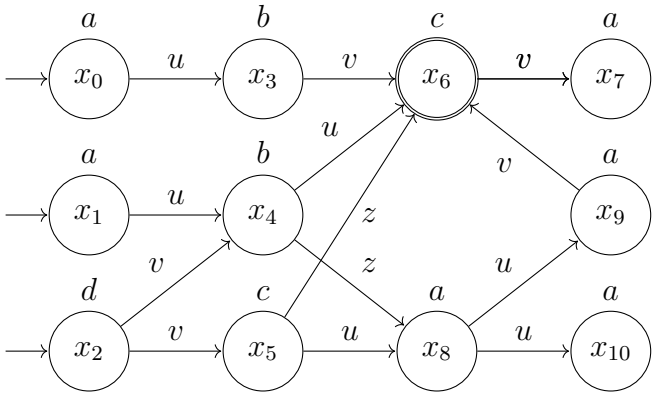

Figure 5.2: System S of Example 12.

order to reach x_6, a controller $C(\beta_1)$ providing a single input is not sufficient. By setting instead $C(\beta_1) = v\,u$, state x_6 is reached starting from either x_3 or x_4. Note that, by applying $C(\beta_1) = u\,v$, state x_6 may be not reached; indeed $f^{ext}(\{x_3\}, u\,v) = \{x_6\}$ whereas $f^{ext}(\{x_4\}, u\,v) = \{x_7\}$. Hence, the order of control inputs in the feedforward controller plays a role. Also the output dynamic feedback feature of the controller is an important aspect. Suppose to start from state x_2; corresponding behaviour is $\beta_0 = d$. By picking feedforward controller $C(\beta_0) = v\,z$, system S can reach either state x_8 or marked state x_6; the same reasoning applies when considering $C(\beta_0) = v\,u$. On the other hand, by choosing $C(d) = v$, $C(d\,v\,b) = u$ and $C(d\,v\,c) = z$, desired state x_6 is indeed reached.

5.3 Main results

We start by giving the following

Definition 38 *Given systems $S_a = (X_a, X_{a,0}, U_a, f_a, X_{a,m}, \Gamma_a, Y_a, H_a)$ and $S_b = (X_b, X_{b,0}, U_b, f_b, X_{b,m}, \Gamma_b, Y_b, H_b)$, S_a is a subsystem of S_b if:*

- $X_a \subseteq X_b$;

- $X_{a,0} \subseteq X_{b,0}$;
- $U_a \subseteq U_b$;
- $f_a(x,u) = f_b(x,u)$ *for all* $x \in X_a$ *and* $u \in \Gamma_a(x)$;
- $X_{a,m} \subseteq X_{b,m}$;
- $\Gamma_a(x) \subseteq \Gamma_b(x)$ *for all* $x \in X_a$;
- $Y_a \subseteq Y_b$;
- $H_a(x) = H_b(x)$ *for all* $x \in X_a$.

The notion above is stronger than the classical notion of subsystems in Definition 23. Indeed, our condition on transition function f_a requires that states reached by S_a, starting from a given state x and with any input $u \in \Gamma_a(x)$, are the same as those of S_b again starting from x with input u.

A state x of S is accessible if there exists a state $x_0 \in X_0$ and a string of inputs $\widehat{u} \in U^*$ such that $x \in f^{ext}(x_0, \widehat{u})$. We denote by $X_{Ac}(S)$ the set of accessible states of S.

Definition 39 *The accessible part of a nondeterministic system S, denoted $NAc(S)$, is the subsystem of S with $X_{Ac}(S)$ as the set of states and containing all and only transitions of S among states in $X_{Ac}(S)$.*

System S is accessible if $S = NAc(S)$. For example, systems in Fig. 5.1 and Fig. 5.2 are accessible.

A state x of S is coaccessible if there exists a state feedback control $\Gamma_z : X \to 2^U$ such that starting from x, the state of S reaches the marked states set in a finite number of transitions.

Definition 40 *The coaccessible part of a nondeterministic system S is the subsystem of S, denoted by $NCoAc(S) = (Z, Z_0, U, f_z, X_m, \Gamma_z, Y, H_z)$, whose entities are computed as follows:*

- *Step 1: Set $Z := X_m$.*
- *Step 2: $\widehat{Z} = \{x \in X \setminus Z \mid \exists u \in U \text{ s.t. } f(x,u) \subseteq Z\}$; $f_z(x,u) := f(x,u)$ for all $x \in \widehat{Z}$ and $u \in U$ such that $f(x,u) \subseteq Z$.*
- *Step 3: $Z := Z \cup \widehat{Z}$. While the cardinality of Z increases, go to Step 2.*
- *Step 4: If $X_0 \subseteq Z$ then $Z_0 := X_0$ otherwise $Z_0 := \varnothing$. Set $\Gamma_z := \{u \in U \mid f_z(x,u)$ is defined$\}$ and $H_z(x) := H(x)$ for all $x \in Z$.*

System S is coaccessible if $S = NCoAc(S)$. The following result shows that the length of behaviours of $NCoAc(S)$ is bounded.

Proposition 8 *Given a system S, there exists $k_{max} \in \mathbb{N}$ such that*[1] *$N(\beta) \leq k_{max}$ for all behaviours β of $NCoAc(S)$.*

Proof: Set $W_1 := X_m$. For all $j > 1$, set W_j as the collection of all $x \in X \setminus \bigcup_{k=1,\ldots,j-1} W_k$ for which there exists $u \in U$ such that $f(x,u) \subseteq \bigcup_{k=1,\ldots,j-1} W_k$. We have:

(i) Since X is finite and $W_j \cap W_{j'} = \varnothing$ for all $j \neq j'$, the sequence of W_j is finite.

(ii) By Step 2 of Definition 40, $f(x,u) \subseteq \bigcup_{k<j-1} W_k$, for all $j > 1$, $x \in W_j$ and $u \in \Gamma_z(x)$.

As a consequence of (i) and (ii), the length of the evolutions in $NCoAc(S)$ is finite and upper bounded by

$$J = \max_{Z_0 \cap W_j \neq \varnothing} j.$$

[1] Notation $N(\beta)$ is introduced after Definition 36.

By definition of behaviour, we get $k_{max} = 2\,J + 2$. ■

After the result of Proposition 8 one might be tempted to say that the $NCoAc$ operator selects the shortest path reaching a marked state. More formally:
Given a system S, for each behaviour β of $NCoAc(S)$ and for any behaviour β' of S such that $X(\beta) = X(\beta')$ the inequality $|\beta| \leq |\beta'|$ holds.

That is not true. Consider the system S in Figure 5.3, consider $\beta = a\,v\,c\,u\,b$ of $NCoAc(S)$, there exists a $\beta' = a\,u\,b$ of S such that $X(\beta) = X(\beta') = \{x_2\}$ but $|\beta| \not\leq |\beta'|$.

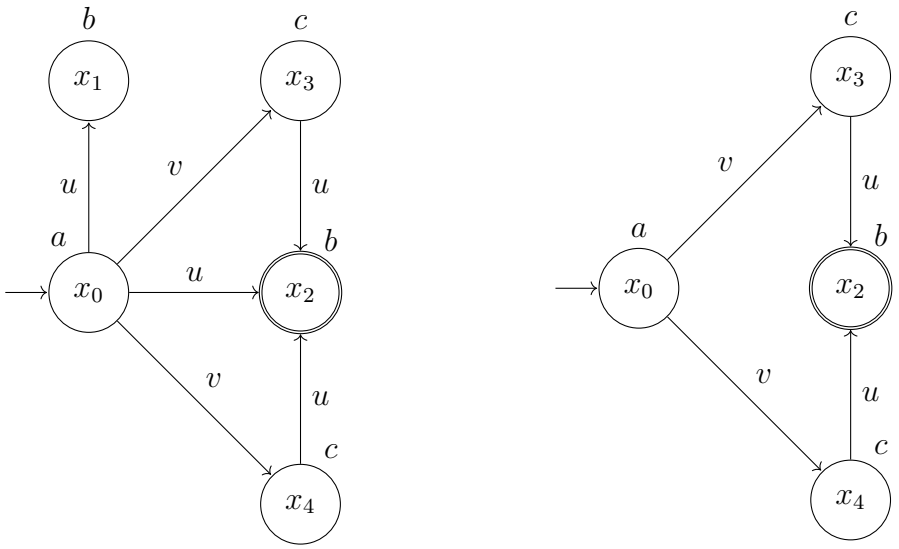

Figure 5.3: System S in the left hand side, system $NCoAc(S)$ in the right hand side.

With this example we have shown that $NCoAc(S)$ does not consider the shortest behaviours of S but *the shortest behaviours that for sure reach a marked state*. In order to formalize the previous sentence we introduce another kind of language that extend the concept of evolution in Definition 9.

Definition 41 *Given a system S as in Definition 7, consider any evolution $\{x_i\}_{i=0,\ldots,n}$ and any input run $\{u_i\}_{i=1,\ldots,n}$ strictly associate to the evolution in S.*

The state-input language is

$$\mathcal{L}^{XU}(S) := \{x_0\, u_1\, x_1\, u_2\, \ldots\, u_n\, x_n \mid \forall\, n \in \mathbb{N}\}$$

The marked state-input language is

$$\mathcal{L}_m^{XU}(S) := \{s \in \mathcal{L}^{XU}(S) \mid s(end) \in X_m\},$$

With the notions above, transition paths are expressed in a sequence, and hence could be considered as a string. We now introduce the concept of related transition paths. They are those having same prefix and at the end of the longest common prefix they evolve differently but with the same input.

Definition 42 *Let s_1 and s_2 be two transition paths in $\mathcal{L}^{XU}(S)$. The transition path s_1 is said to be related to s_2, represented as $s_1 \sim s_2$, if $max\{|p| \mid p \in (\overline{s_1} \cap \overline{s_2}) \setminus \{s_1, s_2\}\}$ is even. Otherwise, they are non-related.*

We now extend the definition above to sets.

Definition 43 *A set $A \subseteq \mathcal{L}^{XU}(S)$ is said to be*

- *related if for any evolution $s_1 \in A$, there exists an $s_2 \in A$ such that $s_1 \sim s_2$, and*
- *maximally related, if there is no such evolution $s_0 \in \mathcal{L}^{XU}(S) \setminus A$ such that $A \cup \{s_0\}$ is related.*

We need this definition to collect all those transition paths that could happen with the same control strategy. In fact we can choose the appropriate input but not the desired transition due to nondeterminism. By Definition 42, it is readily seen that singleton sets are related. Also, every of such related set is maximal whenever the

addition of any other evolution does not preserve its property of being related. We next consider some collections of maximally related sets.

Definition 44 *Given a system S and $NCoAc(S)$, denote*

- $\mathcal{A}_{X_0} := \{A \subseteq \mathcal{L}_m^{XU}(S) \mid A \text{ is maximally related}\}$;
- $\mathcal{A}_{Z_0} := \{A \subseteq \mathcal{L}_m^{XU}(NCoAc(S)) \mid A \text{ is maximally related}\}$.

We point out that $\mathcal{A}_{Z_0} \subseteq \mathcal{A}_{X_0}$, in fact the maximally related concept is captured by Step 2 of Definition 40 where f_z considers all the transitions of f with the same input label. It is easy to very this fact on the systems in Figure 5.3 where $\mathcal{A}_{X_0} = \mathcal{A}_{Z_0} = \{\{x_0\, v\, x_3\, u\, x_2, x_0\, v\, x_4\, u\, x_2\}\}$. The maximally related set $\{x_0\, u\, x_1, x_0\, u\, x_2\}$ is not considered in $\mathcal{A}_{X_0}$ because x_1 is not a marked state.

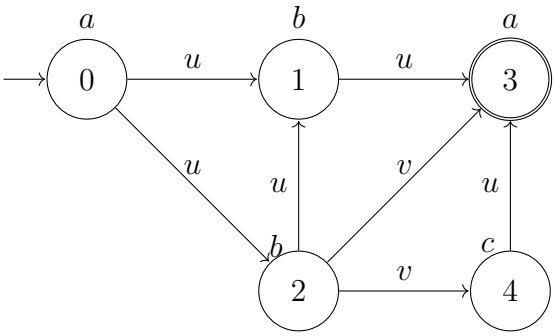

Figure 5.4: System S of Example 13.

Let's illustrate the above notions better with the next example.

Example 13 *Consider the system S in Figure. 5.4, its state-input language is as follows. $\mathcal{L}_m^{XU}(S) = \{0\,u\,1\,u\,3,\ 0\,u\,2\,v\,3,\ 0\,u\,2\,u\,1\,u\,3,\ 0\,u\,2\,v\,4\,u\,3\}$. In this case the string $0\,u\,1\,u\,3$ is related with all the other strings in $\mathcal{L}_m^{XU}(S)$, hence, $\mathcal{A}_{X_0} = \{A\}$ is a singleton with $A = \mathcal{L}_m^{XU}(S)$ (and thus maximally related). Moreover, it is evident that $S = NCoAc(S)$, therefore $\mathcal{A}_{X_0} = \mathcal{A}_{Z_0}$.*

Based on the introduced notations and definitions of transition paths which excludes the output, we now propose an assertion. The claim revolves around $NCoAc(S)$ for a given system S, which actually has the full knowledge of its states, thus permitting us to exclude the output in the form of evolutions introduced in the languages. We next propose that the shortest of the longest possible evolutions of S, that are generated by a controller which ensures reaching the marked states, is found in its coaccessible.

Proposition 9 *For every*

$$A \in \arg \min_{\widehat{A} \in \mathcal{A}_{X_0}} \max_{s \in \widehat{A}} |s|,$$

there exists $A' \in \mathcal{A}_{Z_0}$*, such that* $A = A'$.

Proof: We start considering the case $NCoAc(S)$ is empty. This means that it is impossible to find a control strategy that leads to a marked state; namely, a maximally related set $A \subseteq \mathcal{L}_m^{X\,U}(S)$ does not exists in $\mathcal{A}_{X_0}$. Thus the proposition holds, in fact $\mathcal{A}_{Z_0} = \varnothing$ because $NCoAc(S)$ is empty and $\arg\min_{\widehat{A} \in \mathcal{A}_{X_0}} \max_{s \in \widehat{A}} |s|$ is empty because $\mathcal{A}_{X_0} = \varnothing$ for the discussion above.

Now, consider

$$\mathbf{B} = \arg \min_{\widehat{A} \in \mathcal{A}_{X_0}} \max_{s \in \widehat{A}} |s|.$$

We know that $\mathcal{A}_{Z_0} \subseteq \mathcal{A}_{X_0}$ but we need to prove that $\mathbf{B} \subseteq \mathcal{A}_{Z_0}$.

We can rewrite $\mathbf{B} = \{A_j^{x_0} \mid x_0 \in X_0 \wedge j \in \mathbb{N}\}$, where $A_j^{x_0}$ is a maximally related set in $\mathcal{A}_{X_0}$ with the following properties:

- for all j and for any $s_1, s_2 \in A_j^{x_0}$ we get $s_1(1) = s_2(1) = x_0$;
- by Definition 43 we get: for any $A_j^{x_0}, A_k^{x_0} \in \mathbf{B}$, if $j \neq k$ then $A_j^{x_0} \cap A_k^{x_0} = \varnothing$;

- for any $A_j^{x_0}, A_k^{x_0'} \in \mathbf{B}$, if $x_0 \neq x_0'$ then $A_j^{x_0} \cap A_j^{x_0'} = \varnothing$;
- for any $A_j^{x_0}, A_k^{x_0'} \in \mathbf{B}$ we have

$$\max_{s \in A_j^{x_0}} |s| = \max_{s \in A_k^{x_0'}} |s|$$

Now, suppose that $A_j^{x_0} \notin \mathcal{A}_{Z_0}$ for some $j \in \mathbb{N}$ and $x_0 \in X_0$. By Definition 41 we have $\mathcal{L}_m^{XU}(NCoAc(S)) \subseteq \mathcal{L}_m^{XU}(S)$, it means that there exists a maximally related set $A' \in \mathcal{A}_{Z_0}$ such that $A' \subseteq A_j^{x_0}$. This implies that

$$\max_{s \in A'} |s| \leq \max_{s \in A_j^{x_0}} |s|.$$

But we know $\mathcal{A}_{Z_0} \subseteq \mathcal{A}_{X_0}$, so we have two cases:

1) $\max_{s \in A'} |s|$ is strictly less than $\max_{s \in A_j^{x_0}} |s|$ then A' must be in $\mathbf{B}$ and not $A_j^{x_0}$;
2) $A' = A_j^{x_0}$ and hence $\mathbf{B} \subseteq \mathcal{A}_{Z_0}$.

Hence, from the above cases we conclude that the proposition holds. ■

It can happen that the NCoAc part of a system has some states that are non reachable. Hence, composition of operators NAc and $NCoAc$ follows.

Definition 45 $NTrim(S) = NAc(NCoAc(S))$.

Operators NAc and $NCoAc$ extend the operators in Definition 20 and 21 for deterministic finite state systems to the nondeterministic case. While operators Ac and $CoAc$ in Definition 20 and 21 commute, our operators do not, as shown in the following:

Example 12 (Cont.) Consider system S shown in Figure 5.2. By Definition

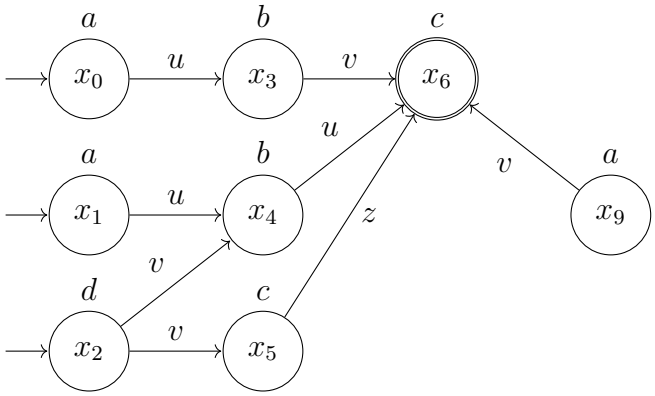

Figure 5.5: System $NCoAc(S)$ of Example 12 (Cont.).

40, it is readily seen that $NCoAc(S)$ coincides with S without states x_7, x_8, x_{10} and without transitions from and to them, as depicted in Figure 5.5. Since state x_9 in $NCoAc(S)$ has no ingoing transitions, $NAc(NCoAc(S))$ coincides with $NCoAc(S)$ without states x_9 and without transitions from x_9 to x_6. Conversely, since S is accessible then $NCoAc(NAc(S)) = NCoAc(S)$. Hence, $NAc(NCoAc(S)) \neq NCoAc(NAc(S))$.

In order to solve Problem 3, we need to associate to system S, a new system that we call *clusterizer* of S and that we denote by $Cl(S)$. States of $Cl(S)$ are called clusters and are subsets of the set of states of S. The set of initial states of $Cl(S)$ is a singleton containing the set of initial states of S. Other clusters of $Cl(S)$ are composed by states of S sharing the same output. Transitions of $Cl(S)$ are labelled with symbols in $U_\varepsilon \times Y_\varepsilon$, detailed as the sets $U \cup \{\varepsilon\}$ and $Y \cup \{\varepsilon\}$, respectively. The set of marked states of $Cl(S)$ is the collection of singletons in 2^{X_m}. Output function is the identity function. The clusterizer $Cl(S)$ extends the notion of observers in Definition 26 for nondeterministic systems, along two directions: first, here S is equipped with an output function; second, here the empty string ε is considered to be measurable for $Cl(S)$. More formally:

Definition 46 *Given a system $S = (X, X_0, U, f, X_m, \Gamma, Y, H)$, the clusterizer of*

S is the system

$$Cl(S) = (X_{Cl}, x_{Cl,0}, U_\varepsilon \times Y_\varepsilon, f_{Cl}, X_{Cl,m}, \Gamma_{Cl}, X_{Cl}, Id)$$

described recursively as follows:

- *Step* 1*:* $X_{Cl} := x_{Cl,0}$*, where* $x_{Cl,0}$ *is a dummy state.*
- *Step* 2*: For any* $y \in Y$ *if the set*

$$\Xi = \{x \in X_0 \mid H(x) = y\} \neq \varnothing$$

  then $X_{Cl} := X_{Cl} \cup \{\Xi\}$ *and* $f_{Cl}(x_{Cl,0}, (\varepsilon, y)) := \{\Xi\}$*.*

- *Step* 3*: For each* $\Xi \in X_{Cl}$*,* $u \in \bigcup_{x \in \Xi} \Gamma(x)$*,* $y \in Y_\varepsilon$*, define:*

$$\begin{aligned} f_{Cl}(\Xi, (u, y)) := \{\{x' \in f(x, u) \mid x \in \Xi \wedge H(x') = y\} \cup \\ \{x \in \Xi \mid f(x, u) = \varnothing \wedge y = \varepsilon\}\}; \\ X_{Cl} := X_{Cl} \cup \{f_{Cl}(\Xi, (u, y))\}. \end{aligned}$$

- *Step* 4*: Repeat Step* 3 *until the entire accessible part of* $Cl(S)$ *has been constructed.*
- *Step* 5*:* $X_{Cl,m} := \{\Xi \in X_{Cl} \mid \Xi = \{x_m\} \subseteq X_m\}$*.*

Since system S is finite, algorithm in the definition above terminates in a finite number of iterations.

Let's see a simple example of application of Cl operator.

Example 14 *Consider system* S *as in Figure 5.6. We construct system* $Cl(S)$ *applying Definition 46.*

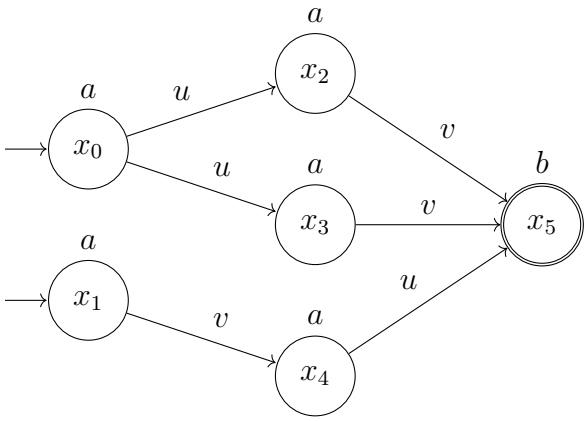

Figure 5.6: System S of Example 14.

First group all initial states according with their output. Hence we have only one cluster state $\{x_0, x_1\}$ *and then connect dummy state* $x_{Cl,0}$ *with the cluster (in this case only one) of initial states* $f_{Cl}(x_{Cl,0}, (\varepsilon, a)) := \{x_0, x_1\}$.

The enabled inputs in cluster $\{x_0, x_1\}$ *are* $\Gamma(x_0) \cup \Gamma(x_1) = \{u, v\}$. *For* $y = a$ *we have* $f_{Cl}(\{x_0, x_1\}, (u, a)) := \{x_2, x_3\}$ *and* $f_{Cl}(\{x_0, x_1\}, (v, a)) := \{x_4\}$. *Instead, a transition from* $\{x_0, x_1\}$ *with* $y = b$ *does not exist. Last but not least, the case* $y = \varepsilon$ *which means that the transition did not happen. Indeed, we have* $f_{Cl}(\{x_0, x_1\}, (u, \varepsilon)) := \{x_1\}$ *because* u *is not enabled in* x_1 *and* $f_{Cl}(\{x_0, x_1\}, (v, \varepsilon)) := \{x_0\}$.

Now, starting from $\{x_2, x_3\}$ *we have only one possibility, input* v. *Hence* $f_{Cl}(\{x_2, x_3\}, (v, b)) := \{x_5\}$.

At this point it is simple to complete the construction of system $Cl(S)$. *For sake of simplicity we avoid considering self-loop with* $y = \varepsilon$, *for instance* $f_{Cl}(\{x_4\}, (v, \varepsilon)) = \{x_4\}$. *The resulting system* $Cl(S)$ *is depicted in Figure 5.7.*

The codomain of f_{Cl} is composed by singletons in the set $2^{X_{Cl}}$; with abuse of notation, we write $f_{Cl}(f_{Cl}(\Xi, (u, y)), (u', y'))$ for all $\Xi \in X_{Cl}$, $u, u' \in U_\varepsilon$ and

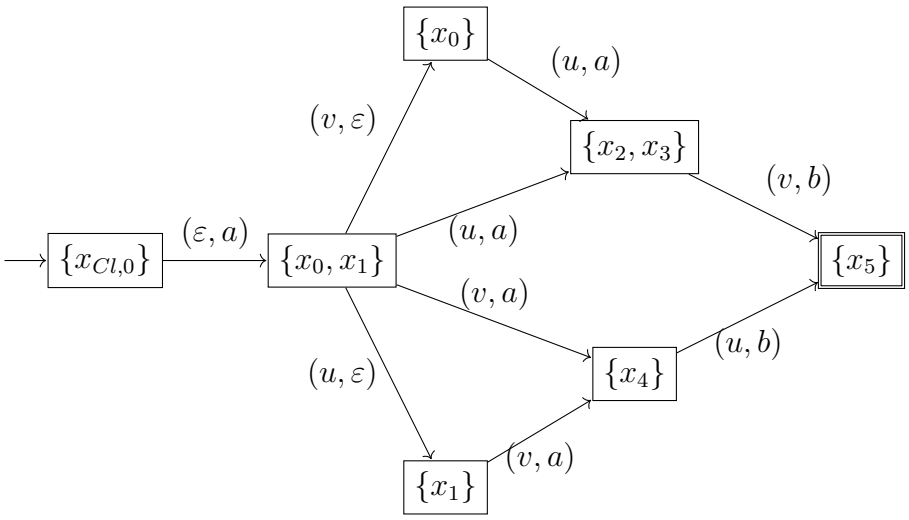

Figure 5.7: System $Cl(S)$ of the system in Figure 5.6.

$y, y' \in Y_\varepsilon$. For later purposes, we extend the partial function f_{Cl} to f_{Cl}^{ext} in the domain $X_{Cl} \times (U_\varepsilon^* \times Y_\varepsilon^*)$, as follows:

$$f_{Cl}^{ext}(\Xi,(u,y)) := f_{Cl}(\Xi,(u,y));$$
$$f_{Cl}^{ext}(\Xi,(\widehat{u}\,u,\widehat{y}\,y)) := f_{Cl}(f_{Cl}^{ext}(\Xi,(\widehat{u},\widehat{y})),(u,y)) \cup f_{Cl}(f_{Cl}^{ext}(\Xi,(\widehat{u},\widehat{y}\,y)),(u,\varepsilon))$$

for any $\widehat{u} \in U_\varepsilon^*, u \in U_\varepsilon, \widehat{y} \in Y_\varepsilon^*$ and $y \in Y_\varepsilon$. Next step consists in defining a system called observer of S and denoted as $Obs(S)$ where non measurable empty strings ε are not present. This corresponds to the so–called determinization of $Cl(S)$. More formally:

Definition 47 *Given the system $Cl(S)$ as in Definition 46, the observer of S is the system*

$$Obs(S) = (X_{Cl}, x_{Cl,0}, U_\varepsilon^* \times Y^*, f_O, X_{Cl,m}, \Gamma_O, X_{Cl}, Id)$$

where f_O is defined as follows:

- *Step* 1*: For any* $\Xi \in X_{Cl}$ *and* $(u, y) \in \Gamma_{Cl}(\Xi)$*, set* $\widehat{u} := u$ *and iteration number* $i := 0$*.*

- *Step* 2*: Set* $Y' := \{\widehat{y} \in Y_\varepsilon^* \mid f_{Cl}^{ext}(\Xi, (\widehat{u}, \widehat{y})) \neq \varnothing\}$*; if the strings in* Y' *are distinguishable and* $\varepsilon \notin Y'$ *then*

$$f_O(\Xi, (\widehat{u}, \widehat{y})) := f_{Cl}^{ext}(\Xi, (\widehat{u}, \widehat{y})),$$

  for all $\widehat{y} \in Y'$ *and go to Step* 1*.*

- *Step* 3*: Set* $i := i + 1$ *and*

$$X_i^{Stop} = \{f_{Cl}^{ext}(\Xi, (\widehat{u}, \widehat{y})) \mid \forall\, \widehat{y} \in Y'\}.$$

- *Step* 4*: If there exists* $j < i$ *such that* $X_i^{Stop} \subseteq X_j^{Stop}$ *then set* $f_O(\Xi, (\widehat{u}, \widehat{y})) := \varnothing$ *and go to Step* 1*.*

- *Step* 5*: For all* $\widehat{y} \in Y'$*,* $(u', y) \in \Gamma_{Cl}(f_{Cl}^{ext}(\Xi, (\widehat{u}, \widehat{y})))$*, update* $\widehat{u} := \widehat{u}\, u'$ *and repeat Step* 2*.*

Since S is finite, algorithm in Definition 47 terminates in finite time. By *Step* 2 above, it is readily seen that

Proposition 10 *Given a system* S*, for any* $\beta \in \mathcal{B}$ *there exists a state* Ξ *in* $Obs(S)$ *such that* $\Xi = X(\beta)$*.*

Let's see a simple example of application of Obs operator.

Example 15 *Consider system* $Cl(S)$ *depicted in Figure 5.7. We construct system* $Obs(S)$ *applying Definition 47.*

Stating from $x_{Cl,0}$ we have $\Gamma_{Cl}(\{x_{Cl,0}\}) = \{(\varepsilon, a)\}$ then we have only one possibility $\widehat{u} = \varepsilon$. The set $\{\widehat{y} \in Y_\varepsilon^* \mid f_{Cl}^{ext}(\{x_{Cl,0}\}, (\varepsilon, \widehat{y})) \neq \varnothing\}$ is equal to $\{a\}$ that is distinguishable hence we can set

$$f_O(\{x_{Cl,0}\}, (\varepsilon, a)) := f_{Cl}^{ext}(\{x_{Cl,0}\}, (\varepsilon, a)) = \{x_0, x_1\}.$$

Now consider state cluster $\{x_0, x_1\}$, $\Gamma_{Cl}(\{x_0, x_1\}) = \{(u, \varepsilon), (v, \varepsilon), (u, a), (v, a)\}$. We have two possibility: $\widehat{u} = u$ or $\widehat{u} = v$. Choose $\widehat{u} = u$

- *The set $\{\widehat{y} \in Y_\varepsilon^* \mid f_{Cl}^{ext}(\{x_0, x_1\}, (u, \widehat{y})) \neq \varnothing\} = \{\varepsilon, a\}$ because from $\{x_0, x_1\}$ there are two transitions with input u: from $\{x_0, x_1\}$ to $\{x_1\}$ with label ε and from $\{x_0, x_1\}$ to $\{x_2, x_3\}$ with label a. Since set $\{\varepsilon, a\}$ is not distinguishable (because $\varepsilon \in \overline{a}$) we must add another input to $\widehat{u}$.*
- *$\Gamma_{Cl}(\{x_2, x_3\}) \cup \Gamma_{Cl}(\{x_1\}) = \{(v, a), (v, b)\}$ hence we update $\widehat{u} = u\,v$. The set $\{\widehat{y} \in Y_\varepsilon^* \mid f_{Cl}^{ext}(\{x_0, x_1\}, (u\,v, \widehat{y})) \neq \varnothing\} = \{a, a\,b\}$, which is an indistinguishable set because $a \in \overline{a\,b}$.*
- *The last sequence of inputs $u\,v\,u$ will generate a distinguishable set $\{a\,b\}$ so we set*

$$f_O(\{x_0, x_1\}, (u\,v\,u, a\,b)) := f_{Cl}^{ext}(\{x_0, x_1\}, (u\,v\,u, a\,b)) = \{x_5\}$$

Similar considerations for $\widehat{u} = v$ where we get

$$f_O(\{x_0, x_1\}, (v\,u\,v, a\,b)) := f_{Cl}^{ext}(\{x_0, x_1\}, (v\,u\,v, a\,b)) = \{x_5\}$$

The resulting system $Obs(S)$ is depicted in Figure 5.8.

We can now present the main result of this chapter.

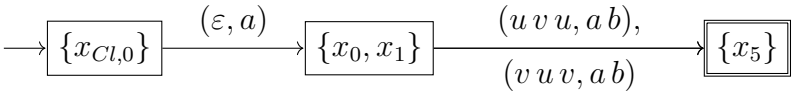

Figure 5.8: System $Obs(S)$ of the system in Figure 5.6.

Theorem 7 *A controller solving Problem 3 exists if and only if* $NTrim(Obs(S))$ *is not empty.*

Proof: (Sufficiency.) Since $NTrim(Obs(S))$ is not empty, a non empty set $\Omega \subseteq X_{Cl,m}$ exists which is accessible in $Obs(S)$. Since

$$f^{ext}(x, \widehat{u}) \subseteq \bigcup_{\widehat{y} \in Y_\varepsilon^*} f_O(\Xi, (\widehat{u}, \widehat{y})),$$

for all $x \in \Xi$ and $(\widehat{u}, \widehat{y}) \in \Gamma_O(\Xi)$, accessibility of $\{x_m\} \in \Omega$ in $NTrim(Obs(S))$ implies accessibility of $x_m \in X_m$ in S. Hence, since Ω has only singletons and the current state of $NTrim(Obs(S))$ is known, by Proposition 8, the statement holds.

(Necessity.) Let C be a controller solving Problem 3. Let $C' = (X_{c'}, \{X_0\}, U_\varepsilon^* \times Y^*, f_{c'}, X_{c',m}, \Gamma_{c'}, X_{c'}, Id)$ be a system where

- $X_{c'} := \{X(\beta) \in 2^X \mid \beta \in \mathcal{B} \text{ and } C(\beta) \text{ is defined}\}$;
- $f_{c'}(X(\beta), (C(\beta), \widehat{y})) := f_{Cl}^{ext}(X(\beta), (C(\beta), \widehat{y}))$, for all $X(\beta) \in X_{c'}$ and $\widehat{y} \in Y^*$ such that $C(\beta\, C(\beta)\, \widehat{y})$ is defined;
- $X_{c',m} := \{\{x_m\} \in X_{c'} \mid x_m \in X_m\}$. Since controller C solving Problem 3 exists, then $NTrim(C')$ is not empty.

By Proposition 10, $X_{c'} \subseteq X_{Cl}$ and $X_{c',m} \subseteq X_{Cl,m}$; by *Step* 2 of Definition 47, $f_{c'}(X(\beta), (C(\beta), \widehat{y})) = f_O(X(\beta), (C(\beta), \widehat{y}))$, for all $X(\beta) \in X_{c'}$ and $\widehat{y} \in Y^*$ such that $C(\beta\, C(\beta)\, \widehat{y})$ is defined. Hence, by Definition 38, C' is a subsystem

of $Obs(S)$ which, by Definition 45 implies that $NTrim(C')$ is a subsystem of $NTrim(Obs(S))$. Since $NTrim(C')$ is not empty then $NTrim(Obs(S))$ is not empty as well. ■

A direct consequence of the result above is the following

Corollary 1 *Suppose $NTrim(Obs(S))$ not empty and let*

$$NTrim(Obs(S)) = (X_c, \{X_0\}, U_\varepsilon^* \times Y^*, f_c, X_{c,m}, \Gamma_c, X_c, Id).$$

A controller C solving Problem 3 is defined by

$$(C(\beta), \widehat{y}) \in \Gamma_c(X(\beta))$$

for some $\widehat{y} \in Y^*$ and for any $\beta = y_0\, \widehat{u}_1\, \widehat{y}_1\, \ldots$, with $\widehat{u}_k = C\,(\beta_{k-1})$ and $\beta_{k-1} = y_0\, \widehat{u}_1 \ldots \widehat{y}_{k-1}$.

We conclude this section with the following

Example 12 (Cont.) Consider Problem 3 for system S in Figure 5.2. By using Definitions 46, 47 and 45 we computed $NTrim(Obs(S))$ that is shown in Figure 5.9.

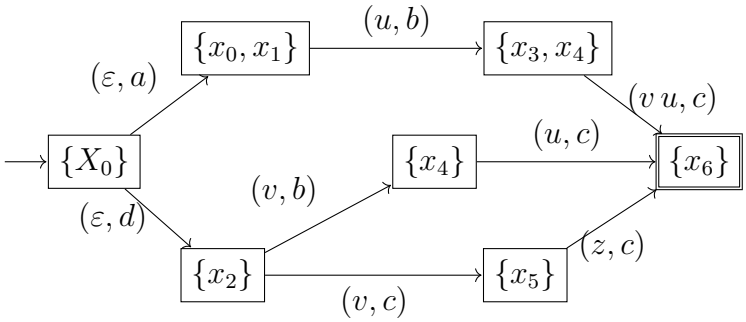

Figure 5.9: System $NTrim(Obs(S))$ of Example 12 (Cont.).

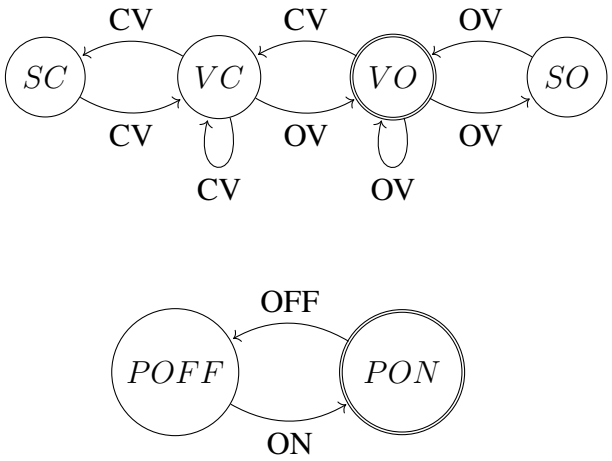

Figure 5.10: Model of the valve (upper panel) and of the pump (lower panel).

Hence, by applying Theorem 7, Problem 3 admits a solution. By applying Corollary 1, controller C solving Problem 3 is given by $C(a) = u$, $C(d) = v$, $C(a\,u\,b) = v\,u$, $C(d\,v\,b) = u$, $C(d\,v\,c) = z$. For instance, from Fig. 5.9, since the only behaviour $\beta = a$ is obtained by the transition from $\{X_0\}$ to $\{x_0, x_1\}$, and the only active input of $\{x_0, x_1\}$ is (u, b), then $C(a) = u$.

5.4 Examples

5.4.1 Pump–valve system

Consider a heating system, called HS, composed of a pump, a valve and a flow sensor. The model of this system has been adapted from (Cassandras & Lafortune, 1999). The pump and the valve are modelled as finite state systems and depicted in Fig. 5.10, where: SC, SO, VC, VO mean Stuck–Closed, Stuck–Open, Valve–Closed, Valve–Open, respectively; labels CV and OV mean Close and Open Valve, respectively; PON and $POFF$ mean Pump–ON and Pump–OFF, respectively. The sensor is modelled as function $Sen : \{VC, VO, SC, SO\} \times$

$\{PON, POFF\} \rightarrow \{N, F\}$, where N and F stand for no flow and flow, respectively. Function Sen returns F only when the valve is open (i.e. it is in states VO and SO) and the pump is on (i.e. it is in state PON). The overall heating system HS is given as the parallel composition (see e.g. (Cassandras & Lafortune, 1999)) of the pump and valve models. Some of the states of S correspond to nominal situations, some others to non nominal ones. For example, state (VO, PON) models a nominal configuration while state (SC, PON) does not. Note that the heating system is nondeterministic; this feature models for example faults. Here we face a typical recovery control problem consisting in designing a controller that steers the state of HS to the nominal configuration (VO, PON). This problem in fact corresponds to solve Problem 3 with $S = HS$ and $X_m = \{(VO, PON)\}$. By applying Theorem 7, this problem admits a solution and by using Corollary 1, we can choose the controller C defined by $C(F) = CV\,OV$ and $C(N) = ON\,CV\,OV\,CV\,OV\,OV\,CV\,OV$. It is possible to see that the strings $F\,C(F)\,\widehat{y}_1$ and $N\,C(N)\,\widehat{y}_2$ belong to $\mathcal{B}$ with $\widehat{y}_1 \in \{F,\ N\,F\}$ and $\widehat{y}_2 \in \{N\,N\,N\,F\,F,\ N\,N\,F\,N\,F\,F,\ N\,N\,F\,F,\ N\,F\,N\,F\,F,\ F\,N\,F\,N\,F\,F,\ F\,F\,N\,F\,F\}$, where the sets involved are distinguishable.

5.4.2 Symbolic control design

In this section we address symbolic control design of a continuous–time nonlinear control system described by the following equations:

$$P : \begin{cases} \dot{x_1}(t) = -2x_1(t) + \tanh(||x_2(t)||) + u(t), \\ \dot{x_2}(t) = -3x_2(t) + 2\,\sin(x_1(t)), \\ y(t) = ||x_1(t)|| - 2\,x_2(t), \\ (x_1(t), x_2(t)) \in \mathbf{X}, (x_1(0), x_2(0)) \in \mathbf{X}_0, \\ u(t) \in \mathbf{U}, y(t) \in \mathbb{R}, t \in \mathbb{R}_0^+, \end{cases} \tag{5.1}$$

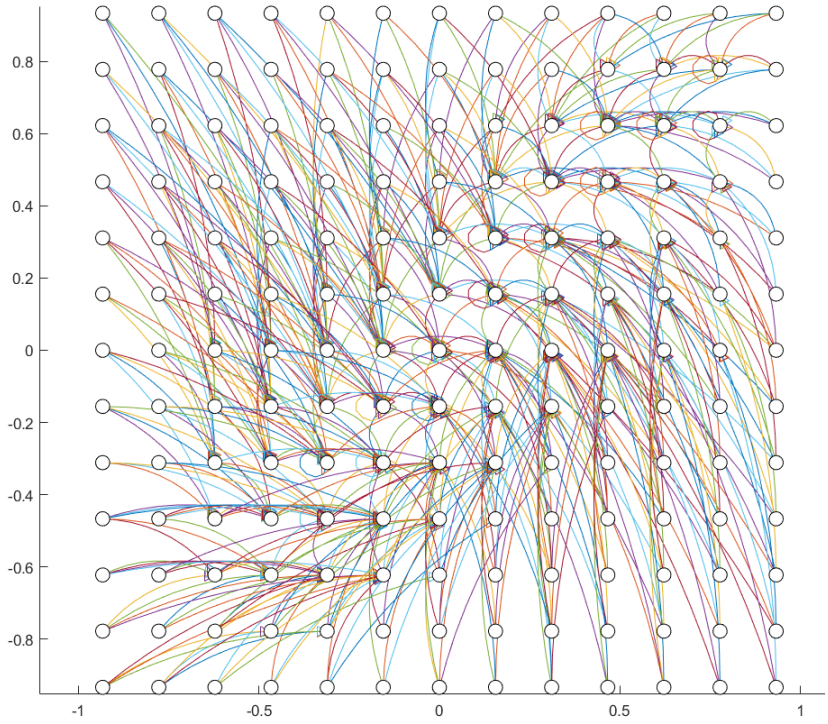

Figure 5.11: System $S_\theta(P)$.

where $||\cdot||$ denotes the infinity norm, $\mathbf{X} = [-1, 1]\times[-1, 1]$, $\mathbf{X}_0 = \{x \in \mathbf{X} \mid ||x|| \geq 0.7\}$ and $\mathbf{U} = \{-0.8, -0.4, 0, 0.4, 0.8\}$. The control problem we address is the following:

Problem 4 *Given the plant P in (5.1), design a controller such that the following specification is met: for any initial state in $\mathbf{X}_0$, the state x of the controlled plant has to cycle, from a certain instant of time on, among points $\omega_1 = (2\eta, 2\eta)$, $\omega_2 = (3\eta, 3\eta)$, $\omega_3 = (\eta, 3\eta)$, $\omega_4 = (\eta, 2\eta)$, in this order, with $\eta = 0.0778$, up to accuracy $\theta = 0.2$ (i.e., infinity norm of difference between state x and ω_i is upper bounded by θ).*

Periodic specification considered in the problem above is relevant and also frequent in the control design of cyber-physical systems and in robotics, see e.g. (Pola & Di Benedetto, 2019; Tabuada, 2009). The solution scheme we use to

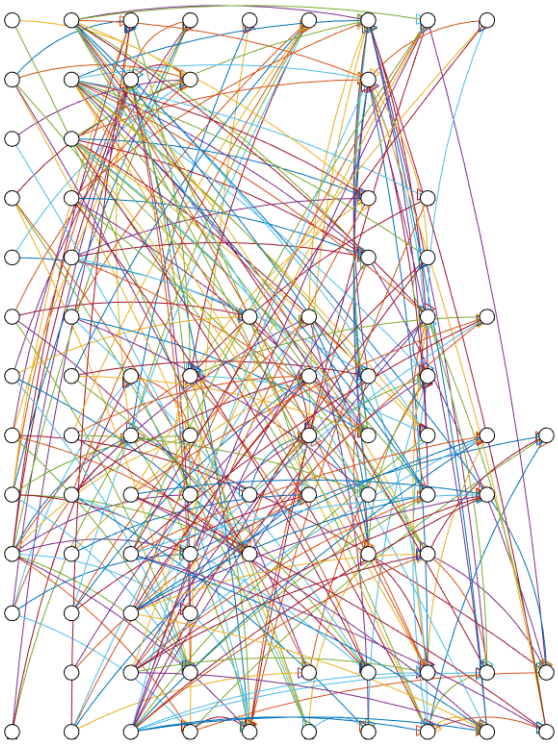

Figure 5.12: System $Trim(Obs(S_\theta(P)))$.

solve Problem 4 consists of *Steps I, II* and *III*, detailed below.

(*Step I:*) By following the results reported in Section 4.3, we construct a finite state system $S_\theta(P)$ that approximates P with accuracy $\theta = 0.2$. System $S_\theta(P)$ is obtained first, by a time discretization of P with sampling time $\tau = 0.3$ and then, by a state space discretization with quantization $\eta = 0.0778$; it is depicted in Figure 5.11 and it consists of 169 states, 88 initial states and 845 transitions. Since it is possible to show that P enjoys the so–called property of Theorem 2 (incremental Global Asymptotic Stability), Theorem 3 guarantees that P and $S_\theta(P)$ are approximately bisimilar with accuracy θ.

(*Step II:*) By applying the results reported in Section 4.3, we design controller C_2 that is assumed to know the state of $S_\theta(P)$, and we find the set $\mathcal{X}_0 = \{-2\eta, 0, 2\eta\} \times \{-2\eta, 0, 2\eta\}$, such that, starting from any initial state in $\mathcal{X}_0$, system $S_\theta(P)$ con-

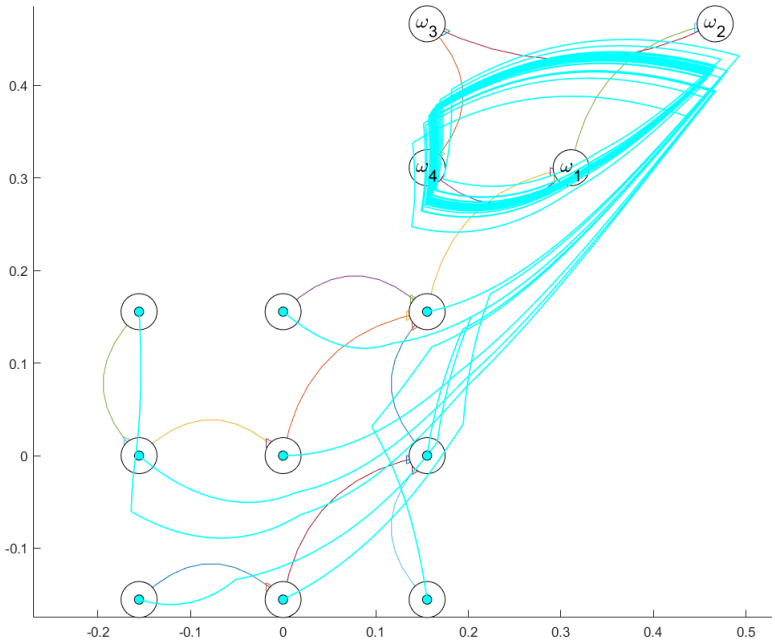

Figure 5.13: Simulations on the controlled plant P.

trolled by C_2 satisfies the desired specification.

(*Step III:*) By using the results presented in the previous sections, we design output feedback controller C_1 which steers any initial state of $S_\theta(P)$ to the set $\mathcal{X}_0$. More precisely, we solve Problem 3 with $S = S_\theta(P)$, $C = C_1$ and $X_m = \mathcal{X}_0$. Resulting observer $Obs(S)$ consists of 108 states, 848 transitions and 9 marked states and resulting $NTrim(Obs(S))$, depicted in Figure 5.12 and it consists of 86 states, 391 transitions and 9 marked states.

The solution to Problem 4 can then be easily obtained by combining controllers C_1 and C_2. First, controller C_1 is applied; once the set $X_m = \mathcal{X}_0$ is reached, controller C_2 is applied. Since P and $S_\theta(P)$ are approximately bisimilar with accuracy θ, then P satisfies the desired specification up to θ. Figure 5.13 shows that trajectories (depicted in cyan) of P starting from any state in $\mathcal{X}_0$ (depicted with cyan dots), satisfy the specification; the underlying finite state system represents controller C_2. The overall time of computation for solving Problem 4 is $22,341$

s. Computations have been performed on a Matlab suite, by using a laptop with CPU Intel Core i7-6700HQ at 2.60 GHz.

5.5 Safety problem formulation

In this section we extend the control problem in Problem 3 to the case of reach-avoid specifications, i.e. specifications where the goal is to reach a target set while avoiding the obstacle set. To this end we now introduce another class of states called *obstacles*. The set of obstacles is denoted by $O \subset X$. They are assumed to be intermediate states, which are neither initial states nor marked states, i.e. $X_0 \cap O = \varnothing$ and $O \cap X_m = \varnothing$; any state $x \in O$ is denoted with a subscript Os i.e. x_{Os}. The aim here is to first incorporate the specification of obstacle avoidance into the reachability control problem considered in the previous section.

Definition 48 *Let S be a system with obstacle states set $O \subset X$. The set $\mathcal{B}^O$ is the collection of behaviours $\beta = y_0\, \hat{u}_1\, \hat{y}_1\, \ldots\, \hat{u}_k\, \hat{y}_k$ of S such that set $Y' = \{\hat{y} \in Y_\varepsilon^* \mid y_0\, \hat{u}_1\, \hat{y}_1 \ldots \hat{u}_k\, \hat{y} \in \mathcal{B}\}$ is distinguishable, and for all $j = 1, \ldots, k$, it holds that $\{\hat{x}_j(i) \mid 1 \leq i \leq |\hat{x}_j|\} \cap O = \varnothing$. Set $X(\beta)$ contains all ending states of evolutions associated to a behaviour $\beta \in \mathcal{B}^O$ of S.*

The definition above selects all those behaviours of S with no associated trajectory that contains a state in the obstacle set O. The control problem we face in this section is then formalized as follows.

Problem 5 *Given a system S, with some obstacle states $O \subset X$, find conditions for the existence of a controller described by function $\widehat{C} : \mathcal{B}^O \to U^*$ and $\mathbf{k} \in \mathbb{N}$, such that for any $\beta = y_0\, \hat{u}_1\, \hat{y}_1\, \cdots \in \mathcal{B}^O$, with $\hat{u}_k = \widehat{C}\,(\beta_{k-1})$ and $\beta_{k-1} =$*

$y_0\,\hat{u}_1\,\hat{y}_1 \ldots \hat{u}_{k-1}\,\hat{y}_{k-1}$, *there exists* $\widehat{k} \leq \mathbf{k}$ *such that*

$$X(\beta_{\widehat{k}}) = \{x_m\} \ \wedge \ x_m \in X_m.$$

It should be noted that $\mathcal{B}^O$ does not always capture the prefixes of its behaviours. However, restricting $\widehat{C}$ to $\mathcal{B}^O$ in the problem statement makes it encompasses the reach-avoid specification. Also, reachability of states in the target set is enforced as done in the previous section, with the use of feedforward and output feedback control strategies. The following example illustrates the definitions introduced above.

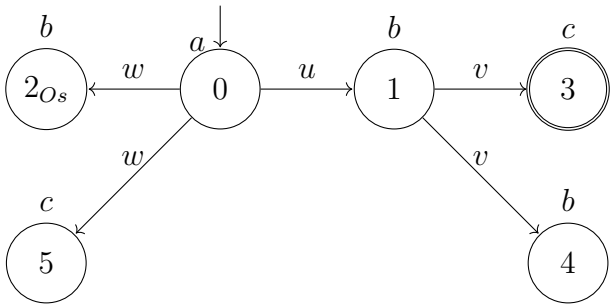

Figure 5.14: System S of Example 16.

Example 16 *The picture in Fig. 5.14 describes a system* S *as in Definition 7, where* $O = \{2\}$. *By Definition 9, string* $0\,1\,3$ *is an evolution of* S*; indeed, a possible input string associated with* 0 *is* $v\,v\,u$*, with* 1 *is* $u\,v$ *and with* 3 *is* ε. *Consider string* $\beta = y_0\,\hat{u}_1\,\hat{y}_1\,\hat{u}_2\,\hat{y}_2$ *with* $y_0 = a$, $\hat{u}_1 = v\,v\,u$, $\hat{y}_1 = b$, $\hat{u}_2 = u\,v$ *and* $\hat{y}_2 = c$. *By Definition 36, string* β *is a behaviour associated with evolution* $0\,1\,3$ *with partition* $\{0, 1, 3\}$. *We now discuss Definition 48. Consider behaviour* $\beta = a\,v\,v\,u\,b\,u\,v\,c$ *defined above. Resulting set* Y' *associated with* $a\,v\,v\,u\,b\,u\,v$ *is* $\{b, c\}$. *By Definition 13 ,* Y' *is a distinguishable set, also* $\{0, 1, 3\} \cap O = \varnothing$ *and* $\{0, 1, 4\} \cap O = \varnothing$. *Hence* $\beta \in \mathcal{B}^O$ *and as a consequence,* $X(\beta) = \{3\}$. *Note that*

β is not the unique behaviour associated to $0\,1\,3$, behaviours in $\mathcal{B}^O$ are infinity, in general. Conversely, consider strings $a\,w\,u\,b$ and $a\,w\,u\,c$, which by Definition 36, are behaviours of S. By Definition 48, the resulting set Y' is distinguishable. But $\{0, 2_{Os}\} \cap O \neq \varnothing$, thus neither of the latter behaviours is in $\mathcal{B}^O$.

5.6 Safety problem solution

In this section, we will introduce another system associated with S, that is by devising another unary operator on S. We now formally define the system, $Os(S)$, which is an associated system of S but have been stripped of every obstacles.

Definition 49 *Consider a system S with obstacles $O \subset X$. The obstacle stripper of S is the system*

$$Os(S) = (X_{Os}, X_0, U, f_{Os}, X_{Os,m}, \Gamma_{Os}, Y, H_{Os})$$

generated as follows:

Step 1: Set $X_{Os} := X_0$.

Step 2: For all $x \in X_{Os}$ and $u \in \Gamma(x)$ such that $f(x,u) \cap O = \varnothing$, set $f_{Os}(x,u) := f(x,u)$ and $H_{Os}(x) := H(x)$.

Step 3: Set $X_{Os} := X_{Os} \cup \{f_{Os}(x,u) \mid x \in X_{Os} \ \wedge \ u \in U\}$.

Step 4: While $card(X_{Os})$ increases, restart from Step 2, else proceed to set $X_{Os,m} := X_{Os} \cap X_m$.

The algorithm terminates in finite time being that finite state systems are considered. It is evident that the states $x \in O$ matters as to whether the evolutions of a system S will lead S to X_m or not. Some $x \in O$ could stand as a main ob-

stacle which could completely prevent evolutions initiated in X_0 from reaching X_m. Therefore, the obstacle positioning is capable of making the logic specification not realizable, where in this case the logic specification is always considered a reach-avoid specification. Let's see a couple of examples where we apply the obstacle stripper.

Example 17 *Consider system S depicted in Figure 5.15 and suppose that state x_5 is a state obstacle, i.e. $O := \{x_5\}$.*

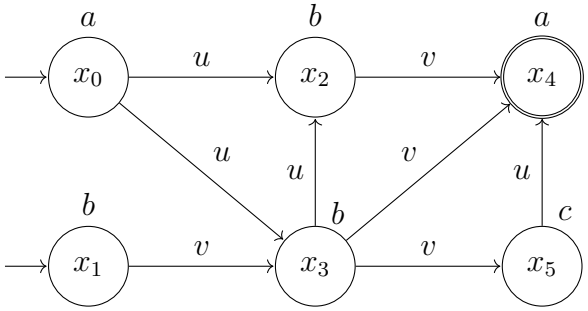

Figure 5.15: System S of Example 17

We construct the obstacle stripper of S in the following manner:

First set $X_{Os} := \{x_0, x_1\}$ the initial states.

For each state in $X_{Os} = \{x_0, x_1\}$ we save the inputs that do not lead to an obstacle state. Since $\Gamma(x_0) = \{u\}$ and $f(x_0, u) \cap O = \varnothing$ we set $f_{Os}(x_0, u) := f(x_0, u) = \{x_2, x_3\}$; $\Gamma(x_1) = \{v\}$ and $f(x_1, v) \cap O = \varnothing$ we set $f_{Os}(x_1, v) := f(x_1, v) = \{x_3\}$. Then we update $X_{Os} := \{x_0, x_1\} \cup \{x_2, x_3\}$.

For x_2 we have $\Gamma(x_2) = \{v\}$ and $f(x_2, v) \cap O = \varnothing$ we set $f_{Os}(x_2, v) := f(x_2, v) = \{x_4\}$. Instead, for x_3 we have $\Gamma(x_3) = \{u, v\}$: $f(x_3, u) \cap O = \varnothing$ hence $f_{Os}(x_3, u) := f(x_3, u) = \{x_2\}$; $f(x_3, v) \cap O = \{x_5\}$ hence we cannot update f_{Os} with input v from x.

The resulting system $Os(S)$ is depicted in Figure 5.16.

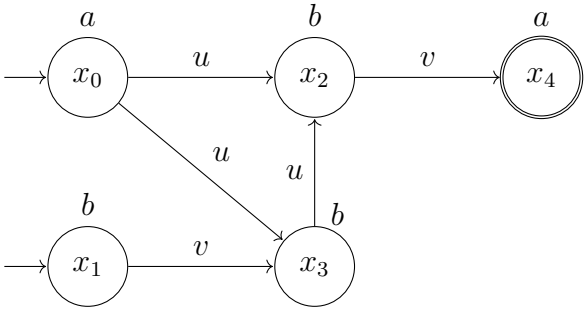

Figure 5.16: System $Os(S)$ of S in Figure 5.15 with $O = \{x_5\}$.

Suppose now that state x_2 is the state obstacle, i.e. $O := \{x_2\}$. From x_0 we must delete all transitions with input u because $f(x_0, u) \cap O = \{x_2\}$. Similarly, from x_3 we must delete the transition with input u because $f(x_3, u) \cap O = \{x_2\}$. The resulting new system $Os(S)$ is depicted in Figure 5.17.

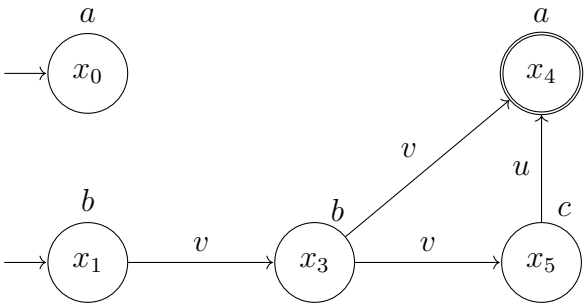

Figure 5.17: System $Os(S)$ of S in Figure 5.15 with $O = \{x_2\}$.

In this second case it is impossible from state x_0 to reach the marked state x_4 so there is no control for this configuration.

Given a system S, the obstacle stripper $Os(S)$ is a subsystem of S. This notion of subsystem is in tandem with Definition 38, which is stronger than the classical notion of subsystem. Also it is evident from Definition 48, that $\mathcal{B}^O$ captures the interesting behaviours for $Os(S)$ as done for S in Definition 36. Moreover, associating to a system S an obstacle stripper $Os(S)$, is a step to solving Problem

5. This arises from our aim of designing a controller that satisfies a more general specification. We recall that Corollary 1 infers the existence of a controller C that solves the control Problem 3 for a system S. Also, by Theorem 7, the existence holds if and only if $NTrim(Obs(S))$ is not empty, i.e always assures reachability specifications, where $NTrim(\cdot)$ and $Obs(\cdot)$ are respectively given by Definition 45 and 47.

Now, consider a system S with obstacles O, which is addressed by Problem 5 in this section. The idea here is to split the desired controller $\widehat{C} : \mathcal{B}^O \rightarrow U^*$ as $\widehat{C} = C \circ C^O$. It is evident by Corollary 1 (complemented with Proposition 10) that if for an obstacle state $x \in O$, there is no states Ξ of $NTrim(Obs(S))$ such that $x \in \Xi$, then the function C^O, that takes the behaviours of S to $\mathcal{B}^O$, is equivalent to Id. This is in tandem with Definition 49, so that if $O = \varnothing$, then $Os(S) = NAc(S)$.

Furthermore, function C^O is well defined over the behaviours of S, provided there are resulting $\beta^O \in \mathcal{B}^O$. Thus, this boils down to the fact that for the existence of C^O for any system S, every corresponding $Os(S)$ generated by Definition 49 is required to be such that $\mathcal{B}^O \neq \varnothing$ and $X_{Os,m} \neq \varnothing$ such that $X(\beta^O) \in X_m$ for some $\beta^O \in \mathcal{B}^O$. Therefore in general, if $O \neq \varnothing$, then $NTrim(Obs(Os(S)))$ not being empty guarantee reachability specifications for $Os(S)$ likewise. However, by definition of operators $NTrim(\cdot)$ and $Obs(\cdot)$ as given in the previous section, the aforementioned conditions on function C^O are being included in the latter general condition. So, we thus improve on the main result of the section 5.2.

Theorem 8 *A controller solving Problem 5 exists if and only if* $NTrim(Obs(Os(S)))$ *is not empty.*

Proof: Since by Definition 49, if $O = \varnothing$, then $Os(S) = NAc(S)$. Other-

wise, $Os(S)$ is a system itself, whose collection of behaviours is a subset of those of S. Hence, Theorem 7 concludes the proof. ■

Similarly, as a corollary to Theorem 5, a controller solving Problem 5 is defined in the system $NTrim(Obs(Os(S)))$.

6 | Data–driven symbolic control

This chapter addresses data-driven control design of an unknown plant, apart from a finite set of experiments, without assuming an a–priori structure of the unknown plant. The proposed controller enforces a desired specifications on the plant up to an accuracy as small as desired. At the end of the chapter an application to the artificial pancreas that shows the potential of the proposed approach. The results of this chapter are based on the works (Pola, Masciulli, De Santis, & Di Benedetto, 2021, 2020).

6.1 Introduction

In monitoring and controlling complex systems, data–driven techniques are becoming increasingly relevant, especially in cases when deriving a mathematical model may be infeasible, for example when the physical system is too complex or in a human-in-the-loop application. System learning and identification are widely used in controller synthesis based on data. With a different approach, controller synthesis can be based directly on data, without assuming an a–priori structure of the unknown model and learning its parametric representation.

In this chapter we follow this second perspective and address data–driven control design where:

- We do not assume a parametric system representation, as e.g. in (Coulson, Lygeros, & Dorfler, 2019), but approach the problem by designing the controller on the basis of the available data.
- We consider a black–box system described by a collection of pairs of input and state functions. We assume that the system is unknown, except for a

set of input-state functions that have been collected and that we call "experiments". The class of systems that may be considered in this framework includes e.g. linear and nonlinear control systems, metric systems, hybrid systems, infinity dimensional systems, and quantized and sampled data control systems where the original plant is a *continuous–time* process.

- The specifications are assumed to be given in terms of a desired regular language defined over an alphabet consisting of a finite set of states of the plant. Regular languages provide a rich framework in the control design of discrete–event, purely continuous and hybrid systems since they are able to represent key specifications such as reachability, safety, motion planning and collision avoidance, periodic orbits, state-based switching, and requirements involving sequences of smaller tasks that need to be performed according to a given order (see e.g. (Tabuada, 2009; Pola & Di Benedetto, 2019)).

Controller design is based only on the knowledge of the collected finite set of experiments, and the proposed controller enforces the specification on the control system, up to an error that can be chosen as small as desired.

Maximality, convergence and adaptivity of the controller as the set of experiments gets bigger are also illustrated. Controller performance on trajectories of the plant different from those in the set of experiments and in the presence of state measurement errors is analyzed. An application related to the artificial pancreas is presented as an illustrative example of our approach.

The literature on data–driven control is vast (see e.g. (Hou & Wang, 2013; Coulson et al., 2019) and the references therein), but this chapter is related principally to (Ghosh, Bansal, Sangiovanni-Vincentelli, Seshia, & Tomlin, 2019; Cheng, Orosz, Murray, & Burdick, 2019; Kazumune & Toshimitsu, 2020; Lavaei, Somenzi,

Soudjani, Trivedi, & Zamani, 2020; Kazemi & Soudjani, 2020; Hashimoto, Saoud, Kishida, Ushio, & Dimarogonas, 2020). In particular, (Ghosh et al., 2019) considers a discrete–time nonlinear control system that is unknown apart from an abstraction of it, with reach–avoid specifications and metrics that quantify the distance between the system and its abstraction. Work (Cheng et al., 2019) considers partially known discrete–time nonlinear control systems with safety specifications and uses reinforcement learning techniques for ensuring safety with high probability during the learning process. Work (Kazumune & Toshimitsu, 2020) focuses on partially known discrete–time nonlinear control systems with safety specifications, and controllers design is based on symbolic systems approximating the original systems. In a stochastic setting, (Lavaei et al., 2020) considers unknown stochastic discrete–nonlinear systems Σ and co–safe LTL specifications; assumptions of Lipschitz continuity of the unknown vector field f, of differentiability of f w.r.t. the unknown noise variable ς and of invertibility of $\partial f/\partial \varsigma$ are made. Work (Kazemi & Soudjani, 2020) considers unknown control Markov processes with continuous state space and shows how reinforcement learning can be applied for computing finite-memory and deterministic sub–optimal policies. Work (Hashimoto et al., 2020) considers partially known discrete–time stochastic nonlinear control systems, and proposes a learning based approach and a safe exploration algorithm, where the trajectory of the system remains in the safe region for all times while collecting the training data and constructing a symbolic abstraction. Apart from the class of specifications that are considered, the main difference with the works described above, except for (Kazemi & Soudjani, 2020), is that we do not make any specific assumption on the *structure* of the discrete–time system describing the unknown plant. Paper (Kazemi & Soudjani, 2020) considers indeed unstructured models though in a stochastic setting (in the form of Markov processes).

6.2 System definition and control problem statement

In this section we first define the abstract system. Then we formulate the control problem here considered. Let $\mathcal{T} = \{(T_1, T_2) \in \mathbb{N} \times \mathbb{N} \mid T_1 < T_2\}$ the set of times. Let $\mathcal{U}$ be the set of input values and $\mathcal{U}^{[T_1;T_2]}$ the set of all input functions $u : [T_1; T_2] \to \mathcal{U}$, with $(T_1, T_2) \in \mathcal{T}$. Let $\mathcal{X}$ be the set of state values and $\mathcal{X}^{[T_1;T_2]}$ the set of all functions $x : [T_1; T_2] \to \mathcal{X}$, denoting the state trajectories. We consider a system described by pairs of input–state functions, as follows:

Definition 50 *An abstract system P is a relation*

$$P \subseteq \bigcup_{(T_1,T_2)\in\mathcal{T}} \left(\mathcal{U}^{[T_1;T_2-1]} \times \mathcal{X}^{[T_1;T_2]}\right) \tag{6.1}$$

satisfying the following properties:

(Suffix closure) If $(u, x) \in P \cap (\mathcal{U}^{[T_1;T_2-1]} \times \mathcal{X}^{[T_1;T_2]})$ for some $(T_1, T_2) \in \mathcal{T}$, then any suffix (u', x') of (u, x) is in P, i.e. $(u', x') \in P \cap (\mathcal{U}^{[T_3;T_2-1]} \times \mathcal{X}^{[T_3;T_2]})$ for any $T_3 \in [T_1; T_2-1]$, where $u'(t) = u(t)$ for any $t \in [T_3; T_2-1]$ and $x'(t) = x(t)$ for any $t \in [T_3; T_2]$;

(Causality) for any $T_3 \in [T_1 + 1; T_2]$ and any $u \in \mathcal{U}^{[T_1;T_2-1]}$, then

$$(P \cap (\{u\} \times \mathcal{X}^{[T_1;T_2]}))|_{[T_1;T_3]} = P \cap (\{u|_{[T_1;T_3-1]}\} \times \mathcal{X}^{[T_1;T_3]}),$$

where $(P \cap (\{u\} \times \mathcal{X}^{[T_1;T_2]}))|_{[T_1;T_3]}$ denotes the collection of pairs $(u, x) \in P \cap (\{u\} \times \mathcal{X}^{[T_1;T_2]})$, where u and x are restricted to the time intervals $[T_1; T_3 - 1]$ and $[T_1; T_3]$, respectively, and $u|_{[T_1;T_3-1]}$ is the restriction of u to $[T_1; T_3 - 1]$;

(Concatenation closure) for any pairs $(u, x) \in P \cap \left(\mathcal{U}^{[T_1;T_2-1]} \times \mathcal{X}^{[T_1;T_2]}\right)$

and $(u', x') \in P \cap (\mathcal{U}^{[T_2;T_3-1]} \times \mathcal{X}^{[T_2;T_3]})$*, for some* $(T_1, T_2), (T_2, T_3) \in \mathcal{T}$*, and satisfying* $x(T_2) = x'(T_2)$*, the pair* $(u'', x'') \in P \cap (\mathcal{U}^{[T_1;T_3-1]} \times \mathcal{X}^{[T_1;T_3]})$*, where* $u''(t) = u(t)$ *for any* $t \in [T_1; T_2 - 1]$*,* $x''(t) = x(t)$ *for any* $t \in [T_1; T_2]$*,* $u''(t) = u'(t)$ *for any* $t \in [T_2; T_3 - 1]$*, and* $x''(t) = x'(t)$ *for any* $t \in [T_2; T_3]$.

This definition follows the general notion of abstract systems given in System Theory, see e.g. (Ruberti & Isidori, 1979). For sake of simplicity, in the sequel we will refer to abstract system as system. Note that causality is necessary to have concatenation closure. We suppose that our control plant is represented by the system P and satisfies the following:

Assumption 5 *System* P *is metric, i.e. we suppose that its set of states* $\mathcal{X}$ *is endowed with a metric* $\mathbf{d} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$.

The system P is assumed to be unknown, apart from a finite set of input-state functions $\mathcal{E}$ contained in P, and collected in some time intervals in $\mathcal{T}$. In the sequel we call elements of $\mathcal{E}$, as experiments. Our aim is to design a controller C for P on the basis of the experiments, in such a way that the controlled systems satisfies some given specifications. Since the time when the controller is designed and applied to P is different from the time when the experiments in $\mathcal{E}$ have been collected, P is supposed to be time–invariant:

Assumption 6 *System* P *is time–invariant, i.e. for any* $(u, x) \in P \cap (\mathcal{U}^{[T_1;T_2-1]} \times \mathcal{X}^{[T_1;T_2]})$ *and any* $t' \in \mathbb{N}$ *we suppose* $(u', x') \in P \cap (\mathcal{U}^{[T_1+t';T_2+t'-1]} \times \mathcal{X}^{[T_1+t';T_2+t']})$*, where* $u'(t) = u(t - t')$ *and* $x'(t) = x(t - t')$ *for any* $t \in [T_1 + t'; T_2 + t']$.

As a consequence, we simplify notations and in the sequel we consider input and state functions in P starting from time 0. Finally, we suppose that

Assumption 7 *System P is deterministic, i.e. for any $u \in \mathcal{U}^{[0;T]}$ with $(0,T) \in \mathcal{T}$ and any $\widehat{x} \in \mathcal{X}$ there exists at most one $x \in \mathcal{X}^{[0;T]}$ such that $x(0) = \widehat{x}$ and $(u,x) \in P$.*

It follows from the definition of system and the assumptions above that P is a discrete–time, time–invariant and deterministic control system. Many classes of discrete–time systems are included in this framework, as e.g. linear, nonlinear and hybrid systems, infinity dimensional systems, or quantized and sampled data control systems where the original plant is a *continuous–time* process. For later purposes, we denote by $\mathcal{X}_0 \subseteq \mathcal{X}$, the set of initial states of P, defined as the collection of all states $x_0 \in \mathcal{X}$ for which there exists a pair $(u,x) \in P$ such that $x_0 = x(0)$. We now define the controller C in the form of a transition system in the sense of Definition 1:

$$C = (X_c, X_{c,0}, U_c, f_c, X_{c,m}, Y_c, H_c). \tag{6.2}$$

The plant P controlled by C is a system as in Definition 50 that we denote by P^C, where the set of state values coincides with the set $\mathcal{X}$ of P and the set of input values coincides with the set U_c of C. Hence, at this level, we can only affirm that $U_c \subseteq \mathcal{U}$ and $Y_c = \mathcal{X}$. Moreover, a pair $(u,x) \in P$ belongs to P^C if u is an input run of C and the state run $\{x_{c,t}\}_{t=0,\dots,T}$ of C associated to u ends in a marked state, i.e. $x_{c,T} \in X_{c,m}$. In the sequel, we will say that C *marks* $u \in \mathcal{U}^{[0;T-1]}$.
We consider as specification a regular language Q defined over a finite alphabet set $\mathbb{X} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the set of states of P. The data–driven controller synthesis problem can be stated as follows:

Problem 6 *Consider a system P, a finite collection of experiments $\mathcal{E} \subseteq P$ and a desired accuracy $\theta \in \mathbb{R}_0^+$. Find a controller C as in (6.2) and a relation $\mathcal{R}_0 \subseteq$*

$\mathcal{X}_0 \times X_{c,0}$, *both depending on the specification* Q *and on the set of experiments* $\mathcal{E}$ *(but not on* P *which is unknown, apart from the set of experiments* $\mathcal{E}$*), which enforce specification* Q *on* P *up to accuracy* θ*, i.e. such that for any pairs* $(u, x) \in P^C \cap (\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]})$*, for some* $(0, T) \in \mathcal{T}$*, with* $(x(0), x_{c,0}) \in \mathcal{R}_0$*, where* $x_{c,0}$ *is the initial state of a state run of* C *marking* u*, there exists a word* $q_0\, q_1 \ldots q_T \in Q$ *such that:*

$$\mathbf{d}(x(t), q_t) \leq \theta,\ \forall t \in [0; T]. \tag{6.3}$$

Remark 4 *From assumptions of suffix and concatenation closures of Definition 50, it follows that for any* $(u_1, x_1), (u_2, x_2) \in \mathcal{E}$*, all suffixes of* (u_1, x_1) *and* (u_2, x_2) *are in* P*; also, the concatenation of* (u_1, x_1) *and* (u_2, x_2)*, when it exists, is in* P*, as well. As a consequence, the set of experiments* $\mathcal{E}$ *can be enlarged with its suffixes and concatenations, which can then be useful for enforcing a part of the specification that is larger than the one only based on* $\mathcal{E}$*. However, in order to simplify notation, in the sequel we will implicitly assume that set* $\mathcal{E}$ *is already suffix and concatenation closed, i.e. it contains all suffixes and concatenations of its elements. This property of* $\mathcal{E}$ *plays a role in deriving the solution to Problem 6, see Remarks 6 and 7.*

Remark 5 *The role of determinism of* P *as in Assumption 7 for solving Problem 6 is discussed hereafter. Suppose for simplicity that the desired accuracy* θ *is set to* 0*. Suppose now that* P *is nondeterministic, i.e. there exist* $(u, x_1), (u, x_2) \in P$ *with* $x_1(0) = x_2(0)$ *and* $x_1 \neq x_2$*. Suppose also that* $(u, x_1) \in \mathcal{E}$ *and* $(u, x_2) \notin \mathcal{E}$ *and that* $x_1 \in Q$ *and* $x_2 \notin Q$*. On the basis of the collection of experiments* $\mathcal{E}$*, one would consider* u *as a control input enforcing the specification because indeed* $(u, x_1) \in \mathcal{E}$ *and* $x_1 \in Q$*. On the other hand, since* P *is nondeterministic, it can evolve starting from initial state* $x_1(0) = x_2(0)$ *and with control input* u*, either with state evolution* x_1 *which is in* Q *or with state evolution* x_2 *which is not only*

not in Q but also unknown (recall $(u, x_2) \notin \mathcal{E}$). Hence, in a nondeterministic setting, it is not possible in general to use a finite collection of experiments to design controllers enforcing regular language specifications.

For later purposes, we give the following

Definition 51 *Let $Q(C, \mathcal{R}_0)$ be the set collecting all words $q_0\, q_1 \ldots q_T \in Q$ for which there exists a pair $(u, x) \in P^C \cap (\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]})$, for some $(0, T) \in \mathcal{T}$, with $(x(0), x_{c,0}) \in \mathcal{R}_0$, where $x_{c,0}$ is the initial state of a state run of C marking u such that (6.3) holds.*

By the definition above, the set $Q(C, \mathcal{R}_0)$ is the part of the regular language specification Q that is enforced by C and $\mathcal{R}_0$ on P. It is readily seen that

Lemma 1 *If $C \sqsubseteq C'$ then $Q(C, \mathcal{R}_0) \subseteq Q(C', \mathcal{R}_0)$. If $\mathcal{R}_0 \subseteq \mathcal{R}_0'$ then $Q(C, \mathcal{R}_0) \subseteq Q(C, \mathcal{R}_0')$.*

6.3 Main result

In this section we provide the solution to Problem 6. To this purpose, we first need to reformulate the specification Q in terms of transition systems, as in Definition 19. Since Q is a regular language, there exists a symbolic transition system $S_Q' = (X_Q', X_{0,Q}', \mathcal{X}, f_Q', X_{Q,m}', Y_Q', H_Q')$, such that its marked input language coincides with Q, i.e., $\mathcal{L}_m^u(S_Q') = Q$. Note that in the definition above the output function can be chosen arbitrarily since it plays no role in ensuring $\mathcal{L}_m^u(S_Q') = Q$. For later purposes, we denote by

$$S_Q = (X_Q, X_{Q,0}, U_Q, f_Q, X_{Q,m}, \mathbb{X}, H_Q) \tag{6.4}$$

the dual symbolic transition system S_Q of system S'_Q, where states of S_Q are transitions of S'_Q and vice versa. Transition system S_Q is metric with metric $\mathbf{d}$.

Next step consists in encoding the experiments in a transition system.

Remark 6 *From Definition 1, it follows that pairs of input and state runs of a transition system naturally satisfy suffix and concatenation closure of Definition 50 (we remind that causality is necessary to have concatenation closure). As a consequence, for the set $\mathcal{E}$ to be encoded in a transition system, pairs $(u, x) \in \mathcal{E}$ need to satisfy suffix and concatenation closure; from Remark 4, this is the case.*

From the discussion above, we can associate to $\mathcal{E}$ the following transition system

$$S(\mathcal{E}) = (X_e, X_{e,0}, U_e, f_e, X_{e,m}, Y_e, H_e), \tag{6.5}$$

where:

- X_e is the collection of states $z \in \mathcal{X}$ for which there exists a pair $(u, x) \in \mathcal{E}$ and $t \in \mathbb{N}$ such that $z = x(t)$;
- $X_{e,0} = X_e \cap \mathcal{X}_0$;
- $U_e = \mathcal{U}$;
- $z^+ = f_e(z, v)$, if there exists $(u, x) \in \mathcal{E}$ and $t \in \mathbb{N}$ such that $z = x(t)$, $v = u(t)$ and $z^+ = x(t+1)$;
- $X_{e,m} = X_e$;
- $Y_e = \mathcal{X}$;
- $H_e(x) = x$ for any $x \in X_e$.

Proposition 11 *Transition system $S(\mathcal{E})$ is: i) deterministic; ii) symbolic; iii) metric, with metric $\mathbf{d}$ for any $x, x' \in X_e$.*

Proposition 12 *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be finite subsets of P. If $\mathcal{E}_1 \subseteq \mathcal{E}_2$, then $S(\mathcal{E}_1)$ is a sub–transition system of $S(\mathcal{E}_2)$, i.e. $S(\mathcal{E}_1) \sqsubseteq S(\mathcal{E}_2)$.*

In order to solve Problem 6, we need to select transitions of $S(\mathcal{E})$ that match transitions of the specification Q up to accuracy θ. To this purpose, we define the following transition system:

$$C' = (X'_c, X'_{c,0}, U'_c, f'_c, X'_{c,m}, \mathcal{X}, H'_c), \tag{6.6}$$

where:

- X'_c is the collection of pairs $(x_e, x_Q) \in X_e \times X_Q$ such that $\mathbf{d}(H_e(x_e), H_Q(x_Q)) \leq \theta$, where we recall that θ is the desired accuracy in Problem 6;
- $X'_{c,0} = X'_c \cap (X_{e,0} \times X_{Q,0})$;
- U'_c is the collection of input values $v \in \mathcal{U}$ for which there exists $(u, x) \in \mathcal{E}$ and $t \in \mathbb{N}$ such that $v = u(t)$;
- $(x_e^+, x_Q^+) = f'_c((x_e, x_Q), u)$ if $x_e^+ = f_e(x_e, u)$ and $x_Q^+ = f_Q(x_Q)$;
- $X_{c,m} = X_e \times X_{Q,m}$;
- $H'_c(x_e, x_Q) = H_Q(x_Q)$ for any $(x_e, x_Q) \in X'_c$.

Transition system C' can be viewed as the product composition of $S(\mathcal{E})$ and S_Q in an approximating sense. A similar notion of approximate product composition appeared previously in (Tabuada, 2008, 2009). In the sequel, we may write $S(\mathcal{E}) \times_\theta S_Q$, instead of C' to emphasize the dependence of C' on $S(\mathcal{E})$, S_Q and

θ. Transition system C' is blocking in general. Since the controllers in P^C are required to fulfill condition (6.3), we need to extract from C', a sub–transition system exhibiting nonblocking behaviour. This is accomplished by computing the transition system

$$C = Trim(C'), \tag{6.7}$$

later on specified by the tuple $(X_c, X_{c,0}, U_c, f_c, X_{c,m}, \mathcal{X}, H_c)$, which is indeed nonblocking. Since $X_c \subseteq X_e \times X_Q$ and sets X_e and X_Q are finite, then X_c is a finite set. Since $U_c \subseteq U_c'$ and U_c' is finite, then U_c is a finite set. As a consequence, controller C is finite. We now have all the ingredients to present the main result of this chapter.

Theorem 9 *Controller C in (6.7) and relation $\mathcal{R}_0$ defined as*

$$\mathcal{R}_0 = \{(x, (x_e, x_Q)) \in \mathcal{X}_0 \times X_{c,0} \mid x = x_e\}, \tag{6.8}$$

solve Problem 6.

Proof: Consider a pair $(u, x) \in P^C \cap (\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]})$, for some $(0, T) \in \mathcal{T}$, with $(x(0), x_{c,0}) \in \mathcal{R}_0$. By definition of P^C, input $u \in \mathcal{U}^{[0;T-1]}$ is marked by C, i.e. such that $u(t) = u_t, t \in [0; T-1]$, where sequence $u_t, t \in [0; T-1]$ is composed of control labels in a state run $\{x_{c,t}\}_{t=0,\ldots,T}$ of the controller C, with

$$x_{c,T} \in X_{c,m}. \tag{6.9}$$

By definition of C we have $x_{c,t} = (x_{e,t}, x_{Q,t})$ for any $t \in [0; T]$, where $\{x_{e,i}\}_{i=0,\ldots,T}$ is a state run of $S(\mathcal{E})$ and $\{x_{Q,i}\}_{i=0,\ldots,T}$ is a state run of S_Q. Since $(x(0), x_{c,0}) \in \mathcal{R}_0$ then

$$x(0) = x_{e,0}. \tag{6.10}$$

Consider now a transition $x_{c,1} = f_c(x_{c,0}, u)$ in C, where $x_{c,t} = (x_{e,t}, x_{Q,t})$, $t = 0, 1$. By Definition 4 on $S(\mathcal{E})$ and S_Q we get $x_{e,1} = f_e(x_{e,1}, u_0)$ and $x_{Q,1} = f_Q(x_{Q,0})$. By definition of f_e, there exists $(u', x') \in P \cap (\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]})$ such that $x_{e,0} = x'(0)$, $u_0 = u'(0)$ and $x_{e,1} = x'(1)$. Since by (6.10) we have $x(0) = x_{e,0}$ and by definition of u_0, we have $u_0 = u(0)$, determinism of P in Assumption 7 and determinism of $S(\mathcal{E})$ (see Proposition 11), imply that $x(1) = x_{e,1}$. By following an induction argument it is easy to see that:

$$x(t) = x_{e,t}, \forall t \in [0; T]. \tag{6.11}$$

By definition of the set of states X_C of C we get:

$$\mathbf{d}(H_e(x_{e,t}), H_Q(x_{Q,t})) \leq \theta, \forall t \in [0; T]. \tag{6.12}$$

By definition of H_e, and conditions (6.11) and (6.12) we get:

$$\mathbf{d}(x(t), H_Q(x_{Q,t})) \leq \theta, \forall t \in [0; T]. \tag{6.13}$$

Let $q_t = H_Q(x_{Q,t})$ for all $t \in [0; T]$. If we prove that the sequence

$$q_0\, q_1 \ldots q_T \in Q, \tag{6.14}$$

by comparing (6.3) and (6.13), the result follows. By (6.9), $x_{c,T} = (x_{e,T}, x_{Q,T}) \in X_{c,m}$ and by definition of marked states $X_{c,m}$ in C, we have $x_{Q,T} \in X_{Q,m}$, from which $q_0\, q_1 \ldots q_T \in \mathcal{L}_m^y(S_Q)$. Since by definition of S_Q we have $\mathcal{L}_m^y(S_Q) = \mathcal{L}_m^u(S'_Q) \setminus \{\varepsilon\}$ and by definition of S'_Q we have $\mathcal{L}_m^u(S'_Q) = Q$, condition (6.14) holds. ■

Remark 7 *Controller C in (6.7), solving Problem 6, is derived on the basis of*

transition systems $S(\mathcal{E})$ and S_Q. Since by Remark 6, definition of $S(\mathcal{E})$ requires the set $\mathcal{E}$ to enjoy the properties of suffix and concatenation closures, the proof of Theorem 9 implicitly assumes such properties to hold for all pairs $(u, x) \in P$ (see Definition 50).

A direct consequence of Theorem 9 is that the part of the specification that can be enforced by controller C and the relation of initial states $\mathcal{R}_0$ coincides with the output marked language of C:

Corollary 2 $Q(C, \mathcal{R}_0) = \mathcal{L}_m^y(C)$.

6.4 Maximality, convergence and adaptivity of the solution

In this section, we discuss maximality of the solution of Problem 6, convergence of the solution as the set of experiments increases, and adaptivity of the proposed controller. The controllers in (6.6) and (6.7) will be denoted respectively by $C'(\mathcal{E}, \theta)$ and $C(\mathcal{E}, \theta)$ to point out their dependence on the set of experiments $\mathcal{E}$ and the accuracy θ. Similarly, $\mathcal{R}_0(\mathcal{E}, \theta)$ will denote the relation of initial states in (6.8), where the dependence on $\mathcal{E}$ and θ is specified. We first show that the controller and relation of initial states solving Problem 6 given in Theorem 9 enforce the largest possible part of the specification on the controlled plant.

Proposition 13 *For any controller C'' and relation $\mathcal{R}_0''$ solving Problem 6 with $C = C''$ and $\mathcal{R}_0 = \mathcal{R}_0''$, we have $Q(C'', \mathcal{R}_0'') \subseteq Q(C(\mathcal{E}, \theta), \mathcal{R}_0(\mathcal{E}, \theta))$.*

Proof: Consider any word $q = q_0\, q_1 \ldots q_T \in Q$, with $T > 0$. Following Theorem 9 and definition of $C(\mathcal{E}, \theta)$, word q which meets (6.3) is in $\mathcal{L}_m^y(C(\mathcal{E}, \theta))$. This implies that for any word $q'' \in \mathcal{L}_m^y(C'') \subseteq Q$ we get $q'' \in \mathcal{L}_m^y(C(\mathcal{E}, \theta))$.

Indeed, since C'' solves Problem 6, then q'' meets (6.3) and we get $\mathcal{L}_m^y(C'') \subseteq \mathcal{L}_m^y(C(\mathcal{E},\theta))$. Hence, by Corollary 2, the statement holds. ■

The next result shows monotonicity of our solution to Problem 6 with respect to increasing accuracies.

Proposition 14 *Let $\theta_1, \theta_2 \in \mathbb{R}_0^+$ be a pair of accuracies and let the finite set of experiments $\mathcal{E} \subseteq P$ be given. If $\theta_1 \leq \theta_2$ then*

$$C(\mathcal{E},\theta_1) \sqsubseteq C(\mathcal{E},\theta_2); \tag{6.15}$$

$$\mathcal{R}_0(\mathcal{E},\theta_1) \subseteq \mathcal{R}_0(\mathcal{E},\theta_2); \tag{6.16}$$

$$Q(C(\mathcal{E},\theta_1),\mathcal{R}_0(\mathcal{E},\theta_1)) \subseteq Q(C(\mathcal{E},\theta_2),\mathcal{R}_0(\mathcal{E},\theta_2)). \tag{6.17}$$

Proof: Let here, for $i = 1, 2$, $C_i' = C'$ in (6.6) with $\theta = \theta_i$. By definition of the set of states of C', condition $\theta_1 \leq \theta_2$ implies $C_1' \sqsubseteq C_2'$, which implies $(C_1') \sqsubseteq (C_2')$ from which, (6.15) holds. By (6.8), condition (6.15) implies condition (6.16). Finally by Lemma 1, conditions (6.15) and (6.16) imply (6.17). ■

The intuition behind the result above is that as accuracy parameter increases, our solution to Problem 6 may find more transitions of $S(\mathcal{E})$ that match transitions of S_Q. The next result shows monotonicity of our solution to Problem 6 with respect to increasing sets of experiments.

Proposition 15 *Let $\mathcal{E}_1 \subseteq P$ and $\mathcal{E}_2 \subseteq P$ be a pair of finite collections of experiments on P and let the accuracy $\theta \in \mathbb{R}_0^+$ be given. If $\mathcal{E}_1 \subseteq \mathcal{E}_2$ then*

$$C(\mathcal{E}_1,\theta) \sqsubseteq C(\mathcal{E}_2,\theta); \tag{6.18}$$

$$\mathcal{R}_0(\mathcal{E}_1,\theta) \subseteq \mathcal{R}_0(\mathcal{E}_2,\theta); \tag{6.19}$$

$$Q(C(\mathcal{E}_1,\theta),\mathcal{R}_0(\mathcal{E}_1,\theta)) \subseteq Q(C(\mathcal{E}_2,\theta),\mathcal{R}_0(\mathcal{E}_2,\theta)). \tag{6.20}$$

Proof: By Proposition 12, if $\mathcal{E}_1 \subseteq \mathcal{E}_2$, then $S(\mathcal{E}_1) \sqsubseteq S(\mathcal{E}_2)$, which implies by definition of C' in (6.6) that $C'_1 \sqsubseteq C'_2$, where here, for $i = 1, 2$, we set $C'_i = C'$ in (6.6) with $\mathcal{E} = \mathcal{E}_i$. Since $C'_1 \sqsubseteq C'_2$ implies $(C'_1) \sqsubseteq (C'_2)$, then (6.18) holds. By (6.8), condition (6.18) implies condition (6.19). Finally by Lemma 1, conditions (6.18) and (6.19) imply (6.20). ■

The intuition behind the result above is that as the set of experiments increases (in the sense of sets inclusion), our solution to Problem 6 may find more transitions of P that match transitions of S_Q. The following result establishes convergence properties of our solution to Problem 6.

Proposition 16 *Consider a sequence $\mathcal{E}_{seq} = \{\mathcal{E}_i\}_{i\in\mathbb{N}}$ of finite sets of experiments on P and suppose that $\mathcal{E}_i \subseteq \mathcal{E}_{i+1} \subseteq P$, for any $i \in \mathbb{N}$. Then, there exists $i(\mathcal{E}_{seq}) \in \mathbb{N}$ such that for any $i \geq i(\mathcal{E}_{seq})$:*

$$Q(C(\mathcal{E}_i, \theta), \mathcal{R}_0(\mathcal{E}_i, \theta)) = Q(C(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta), \mathcal{R}_0(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta)). \tag{6.21}$$

Proof: We consider two cases: case i) the cardinality of P is finite and, case ii) the cardinality of P is infinity.

Case i): There exists $i(\mathcal{E}_{seq}) \in \mathbb{N}$ such that $\mathcal{E}_i = \mathcal{E}_{i(\mathcal{E}_{seq})}$ for any $i \geq i(\mathcal{E}_{seq})$. As a consequence, the statement holds.

Case ii): Let $\{\mathcal{E}_j\}_{j\in J}$, with $J \subseteq \mathbb{N}$, be the sequence containing all the sets in $\{\mathcal{E}_i\}_{i\in\mathbb{N}}$ such that there is no $j, j' \in J$, with $j \neq j'$, such that $\mathcal{E}_j = \mathcal{E}_{j'}$. Here, we consider two subcases:

Case ii.1) (the cardinality of J is finite) let $i(\mathcal{E}_{seq})$ be the maximal element of J. By definition of sequence $\{\mathcal{E}_j\}_{j\in J}$, we have that $\mathcal{E}_i = \mathcal{E}_{i(\mathcal{E}_{seq})}$ for any $i \geq i(\mathcal{E}_{seq})$ from which, the statement holds.

Case ii.2) (the cardinality of J is infinity) let $\{\mathcal{E}_k\}_{k\in K}$, with $K \subseteq J$, be the sequence containing all the sets in $\{\mathcal{E}_j\}_{j\in J}$ such that there is no $k, k' \in K$, with $k \neq k'$, such that $C(\mathcal{E}_k, \theta) = C(\mathcal{E}_{k'}, \theta)$. Here, we consider two subcases:

Case ii.2.a) (the cardinality of K is finite)
let $i(\mathcal{E}_{seq})$ be the maximal element of K. We have

$$C(\mathcal{E}_i, \theta) = C(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta), \forall i \geq i(\mathcal{E}_{seq}), \tag{6.22}$$

which by (6.8), implies

$$\mathcal{R}_0(\mathcal{E}_i, \theta) = \mathcal{R}_0(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta), \forall i \geq i(\mathcal{E}_{seq}). \tag{6.23}$$

By combining (6.22) and (6.23) we get the statement.

Case ii.2.b) (the cardinality of K is infinity)
let $X(\mathcal{E}_k, \theta)$, $X_0(\mathcal{E}_k, \theta)$, $f_{\mathcal{E}_k,\theta}$ be the sets of states, the set of initial states and the transition relation of $C(\mathcal{E}_k, \theta)$, respectively. Since X_Q, $X_{Q,0}$ and f_Q are finite sets, by definition of $C(\mathcal{E}_k, \theta)$, for any $k \in K$, cardinality of sets $X(\mathcal{E}_k, \theta)$, $X_0(\mathcal{E}_k, \theta)$ and $f_{\mathcal{E}_k,\theta}$ is upper bounded by cardinality of sets X_Q, $X_{Q,0}$ and f_Q, respectively. By definitions of K we get $C(\mathcal{E}_k, \theta) \sqsubset C(\mathcal{E}_{k'}, \theta)$, with $k, k' \in K$ and $k < k'$, and hence by definition of $\sqsubset$ we have $X(\mathcal{E}_k, \theta) \subset X(\mathcal{E}_{k'}, \theta)$ or $X_0(\mathcal{E}_k, \theta) \subset X_0(\mathcal{E}_{k'}, \theta)$ or $f_{\mathcal{E}_k,\theta} \subset f_{\mathcal{E}_{k'},\theta}$. Since K is an infinity set, a contradiction holds.

■

Intuitively, the result above shows that as the set of experiments gets bigger, there is a step $i(\mathcal{E}_{seq})$ after which the corresponding part of the specification Q enforced

cannot "increase" anymore (in the sense of inclusion in (6.20)). As it is clear from the proof, this is a consequence of the fact that regular languages specifications can be encoded by transition systems that are symbolic, i.e. their sets of states and inputs have finite cardinality. We then obtain the following result:

Corollary 3 *Consider two sequences* $\mathcal{E}_{seq} = \{\mathcal{E}_i\}_{i\in\mathbb{N}}$ *and* $\mathcal{E}'_{seq} = \{\mathcal{E}'_i\}_{i\in\mathbb{N}}$ *of finite sets of experiments on* P *and suppose that*

$$\mathcal{E}_i \subseteq \mathcal{E}_{i+1} \subseteq P, \quad \mathcal{E}'_i \subseteq \mathcal{E}'_{i+1} \subseteq P, \quad \forall i \in \mathbb{N} \tag{6.24}$$

and that

$$\bigcup_{i\in\mathbb{N}} \mathcal{E}_i = P, \quad \bigcup_{i\in\mathbb{N}} \mathcal{E}'_i = P. \tag{6.25}$$

Then,

$$Q(C(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta), \mathcal{R}_0(\mathcal{E}_{i(\mathcal{E}_{seq})}, \theta)) = Q(C(\mathcal{E}'_{i(\mathcal{E}'_{seq})}, \theta), \mathcal{R}_0(\mathcal{E}'_{i(\mathcal{E}'_{seq})}, \theta)).$$

The proof of the corollary above is a direct consequence of Proposition 16, condition (6.25) and Proposition 13 and is therefore omitted. This result shows that independently of the sequences of sets of experiments $\mathcal{E}_{seq}$ and $\mathcal{E}'_{seq}$ and provided that these sets satisfy conditions (6.24) and (6.25), the parts of the specification Q that can be enforced by the corresponding solutions to Problem 6 coincide asymptotically. We conclude this section by showing the adaptivity of the solution to Problem 6 with respect to increasing sequences of experiments.

Proposition 17 $C(\mathcal{E}_1 \cup \mathcal{E}_2, \theta) = (C'(\mathcal{E}_1, \theta) \sqcup C'(\mathcal{E}_2, \theta))$.

Proof: By definition of $S(\mathcal{E})$ in (6.5) we have $S(\mathcal{E}_1 \cup \mathcal{E}_2) = S(\mathcal{E}_1) \sqcup S(\mathcal{E}_2)$ from which, $C'(\mathcal{E}_1 \cup \mathcal{E}_2, \theta) = S(\mathcal{E}_1 \cup \mathcal{E}_2) \times_\theta S_Q = (S(\mathcal{E}_1) \sqcup S(\mathcal{E}_2)) \times_\theta S_Q$. By

definition of approximate product composition operator $\times_\theta$, it is straightforward to check that $(S(\mathcal{E}_1) \sqcup S(\mathcal{E}_2)) \times_\theta S_Q = (S(\mathcal{E}_1) \times_\theta S_Q) \sqcup (S(\mathcal{E}_2) \times_\theta S_Q)$. Hence, since $C'(\mathcal{E}_1, \theta) = S(\mathcal{E}_1) \times_\theta S_Q$ and $C'(\mathcal{E}_2, \theta) = S(\mathcal{E}_2) \times_\theta S_Q$, we get $C'(\mathcal{E}_1 \cup \mathcal{E}_2, \theta) = C'(\mathcal{E}_1, \theta) \sqcup C'(\mathcal{E}_2, \theta)$. Finally, since $C(\mathcal{E}_1 \cup \mathcal{E}_2, \theta) = (C'(\mathcal{E}_1 \cup \mathcal{E}_2, \theta))$, the result follows. ■

6.5 Controller performance on the unknown plant

The results we presented in the previous sections use experiments to design a controller enforcing regular language specifications. In this section we analyze controller performance on trajectories of the plant different from those in the set of experiments and in the presence of state measurement errors. We first assume that if for some $u \in \mathcal{U}^{[0;T-1]}$ and $\widehat{x} \in \mathcal{X}_0$ there exists a pair $(x, u) \in P$ with $x(0) = \widehat{x}$, then for any $\widehat{x}' \in \mathcal{X}_0$ there exists a pair $(x', u) \in P$ with $x'(0) = \widehat{x}'$. The pair (x', u) is unique because of Assumption 7. We also assume the knowledge of a function $\beta : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that $\mathbf{d}(x_1(t+1), x_2(t+1)) \leq \beta(\mathbf{d}(x_1(t), x_2(t)))$, $\forall t \in [0; T]$, for any pair of trajectories $(u, x_1), (u, x_2) \in P \cap \bigcup_{(0,T) \in \mathcal{T}} \left(\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]}\right)$. We denote by β^t the function obtained as the t times composition of function β. Let $\rho \in \mathbb{R}^+$ denote the bound on the state measurement error in the set of experiments. Due to this error, the initial state $x(0)$ of P when the experiment was collected may differ from its measure $z(0)$; consequently, the distance between $x(0)$ and $z(0)$ is upper bounded by ρ, i.e. $\mathbf{d}(x(0), z(0)) \leq \rho$. If x denotes the state trajectory of P starting from initial state $x(0)$ with control u, which exists in view of the first assumption of this section, then $\mathbf{d}(x(t), z(t)) \leq \rho, \forall t \in [0; T]$. Consider an experiment $(u, z) \in \mathcal{E}$, for which there exists $q = q_0\, q_1 \ldots q_T \in Q$ such that $\mathbf{d}(z(t), q_t) \leq \theta$ for all $t \in [0; T]$. Given the measurement z' of the current state trajectory x' with control u, suppose that $\mathbf{d}(z'(0), z(0)) \leq \lambda$.

Then $\mathbf{d}(x'(0), x(0)) \leq 2\rho + \lambda$. Hence, by recalling the definition of β^t we get $\mathbf{d}(x'(t), q_t) \leq \beta^t(2\rho + \lambda) + \rho + \theta, \forall t \in [0; T]$. This means that the accuracy of the controller has decreased by an index whose upper bound can be quantified by the index $p = \sup_{(0,T)\in\mathcal{T}(P^C)} \max_{t\in[1;T]} \beta^t(2\rho + \lambda) + \rho + \theta$, where $\mathcal{T}(P^C)$ denotes the collection of time intervals $(0, T) \in \mathcal{T}$ for which there exists a pair $(u, x) \in P^C \cap (\mathcal{U}^{[0;T-1]} \times \mathcal{X}^{[0;T]})$.

6.6 Application to the artificial pancreas

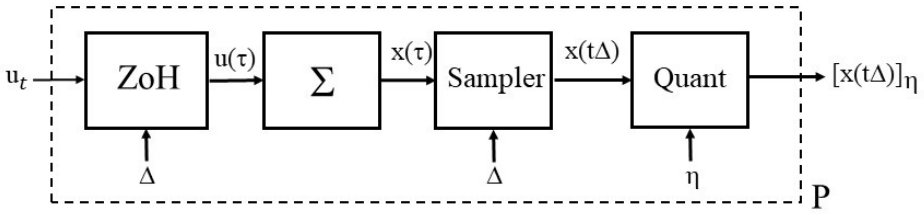

Figure 6.1: Control scheme for the artificial pancreas.

There is a rich literature on the artificial pancreas, see e.g. (Borri, Cacace, et al., 2017) and the references therein, where a data-driven approach is followed e.g. in (Kushner, Bortz, Maahs, & Sankaranarayanan, 2018; Dutta, Kushner, & Sankaranarayanan, 2018). Here we consider this problem as a case study to show the applicability of the general methodology we proposed in the previous sections. The plant P we consider is depicted in Fig. 6.1 and given as the serial interconnection of a Zero order Holder (ZoH), a continuous–time nonlinear control system Σ, modeling the artificial pancreas, a $Sampler$ and a quantizer $Quant$. *We suppose that P is unknown*. From Fig. 6.1 we are thinking to control Σ through quantized and digital controllers. Moreover, we are supposing that state variables are measured by an ideal sensor and are therefore quantized uniformly. The control scheme we consider is described in details, as follows. The ZoH associates to the input run $\{u_t\}_{t=0,\dots,T-1}$ taking values in the finite set

$\mathcal{U} = 4\mathbb{Z} \cup [0, 48]$, the piecewise constant control input $u : [0, T-1] \to \mathcal{U}$ defined by $u(\tau) = u_t, \forall \tau \in [t\Delta, (t+1)\Delta[, \quad t \in [0; T-1]$, where $\Delta = 5\, min$ is the clock of the microprocessor implementing the controller C, as in digital devices. The continuous–time nonlinear control system Σ is taken from (Borri, Palumbo, Manes, Panunzi, & De Gaetano, 2017) and described by:

$$\begin{cases} \frac{dG(\tau)}{d\tau} = -K_{xgi} G(\tau) I(\tau) + \frac{T_{gh}}{V_G}, \\ \frac{dI(\tau)}{d\tau} = -K_{xi} I(\tau) + \frac{T_{iGmax}}{V_I} h(G(\tau)) + u(\tau), \tau \in \mathbb{R}_0^+, \end{cases}$$

where

- G is the glucose concentration in the plasma $[mg/dL]$;
- I is the insulin concentration in the plasma $[pM]$;
- $K_{xgi} = 7.45\, 10^{-5}$ is the rate of glucose uptake by tissues per unit of plasma insulinemia $[pM^{-1}min^{-1}]$;
- $T_{gh} = 0.45$ is the net balance between hepatic glucose output and zero-order glucose tissue uptake $[(mg/KgBW)min^{-1}]$;
- $V_G = 1.3$ is the apparent distribution volume for glucose $[dL/KgBW]$;
- $K_{xi} = 0.1$ is the apparent linear insulin clearance rate $[min^{-1}]$;
- $T_{iGmax} = 1.39$ is the maximal second-phase insulin release rate $[(pmol/KgBW)min^{-1}]$;
- $V_I = 0.24$ is the apparent insulin distribution volume $[L/KgBW]$;
- $h(\cdot)$ is a nonlinear function representing the endogenous pancreatic Insulin Delivery Rate (IDR) and defined as $h(G) = (G/G^*)^\gamma/(1 + (G/G^*)^\gamma)$, where
  - $\gamma = 2.3$ (dimensionless) denotes the progressiveness of the pancreas

reaction to circulating glucose concentrations and

  - $G^* = 162\ [mg/dL]$ is the glucose concentration at which the insulin release reaches half of its maximal rate;

- u is the exogenous intra-venous IDR, which takes the role of control input $[pM/min]$.

In the sequel, we denote by $x(\tau) = (G(\tau), I(\tau))$ the state of Σ at time $\tau \in \mathbb{R}_0^+$. $Sampler$ associates to the continuous signal $x(\cdot)$ the sequence $\{x(t\Delta)\}_{t\in\mathbb{N}}$. Finally, the quantizer $Quant$ associates to the sequence $\{x(t\Delta)\}_{t\in\mathbb{N}}$, the sequence $\{[x(t\Delta)]_\eta\}_{t\in\mathbb{N}}$ defined by $[x(t\Delta)]_\eta = ([G(t\Delta)]_{\eta_G}, [I(t\Delta)]_{\eta_I})$ with $\eta_G = 4.5$ $[mg/dL]$ and $\eta_I = 10\ [pM]$.

The specification Q we consider requires that state variables of P reach in one step a set $\mathbb{T}_{j+1}$ from a set $\mathbb{T}_j$, for $j \in [1; 9]$, where

- $\mathbb{T}_1 = (2\eta_G\,[16; 18]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_2 = (2\eta_G\,[15; 17]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_3 = (2\eta_G\,[14; 16]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_4 = (2\eta_G\,[13; 15]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_5 = (2\eta_G\,[12; 14]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_6 = (2\eta_G\,[12; 13]) \times (2\eta_I\,[0; 6])$,
- $\mathbb{T}_7 = (2\eta_G\,[11; 12]) \times (2\eta_I\,[0; 4])$,
- $\mathbb{T}_8 = (2\eta_G\,[10; 11]) \times (2\eta_I\,[0; 4])$,
- $\mathbb{T}_9 = \{2\eta_G\,10\} \times (2\eta_I\,[0; 4])$,
- $\mathbb{T}_{10} = \{2\eta_G\,10\} \times (2\eta_I\,[0; 2])$.

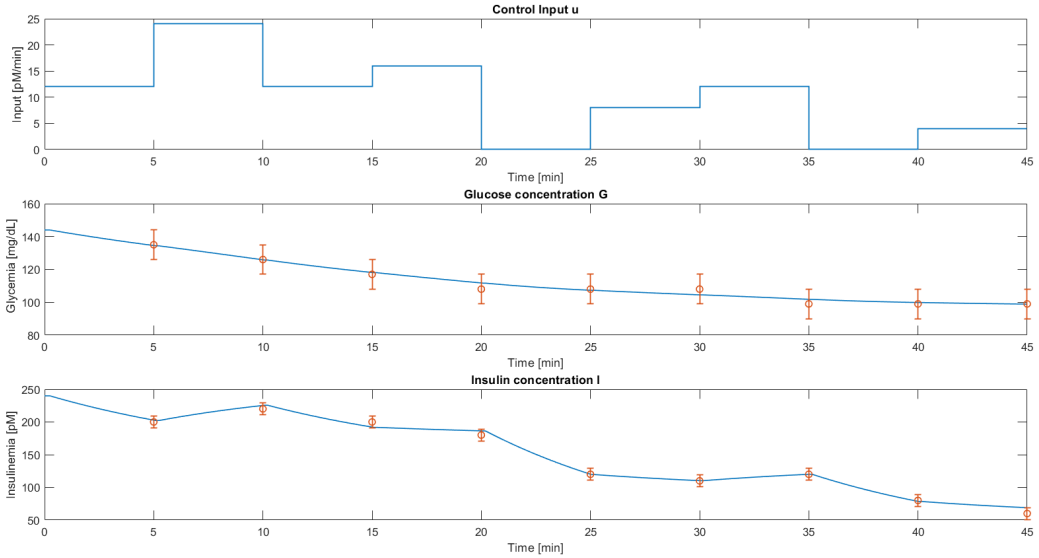

Figure 6.2: In the first panel, a control input enforcing the specification. In the second and third panels, the obtained state trajectory of Σ.

Specification above can be easily formalized in the framework of regular languages, see e.g. (Pola & Di Benedetto, 2019). We encoded the specification Q in a transition system S_Q consisting of 103 states and $1,062$ transitions. The accuracy we consider solving our control problem is $\theta = 9$. By following the results illustrated in the previous sections, we collected $20,000$ experiments, randomly generated by P, with $(0, T) \in \mathcal{T}$ where $T \leq 30$, i.e. we suppose to have no more than 30 measurements per each experiment. We encoded the experiments in the transition system $S(\mathcal{E})$ consisting of 466 states and $311,445$ transitions. Time of computation is $1,444\ s$. We computed the controller $C(\mathcal{E}, \theta)$ enforcing the specification Q on P. Resulting controller consists of 100 states and 659 transitions. Time of computation of $C(\mathcal{E}, \theta)$ is $408\ s$. In Fig. 6.2 we show the controlled plant $P^{C(\mathcal{E},\theta)}$ starting from initial state $x(0) = (144\,[mg/dL], 240\,[pM])$. In the first panel we show the control input. In the second and third panels, we depict controlled trajectory of P and red vertical bars centered at a word of the specification Q and with amplitude θ; from this plot it is easy to see that

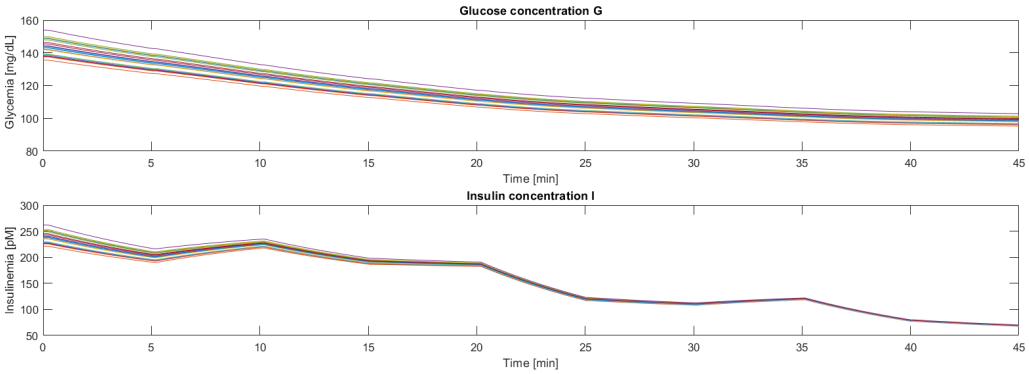

Figure 6.3: Collection of 21 state trajectories of Σ with initial states in the set $[134, 154]\,[mg/dL] \times [220, 260]\,[pM]$ and control input as in the first panel of Fig. 6.2.

the specification Q is met. In Fig. 6.3 we show the controller performance with respect to initial states of the plant that are different from the ones considered in the experiments. We consider 21 state trajectories starting from the set $[134, 154]\,[mg/dL] \times [220, 260]\,[pM]$ of possible initial states and with the same control input as the one depicted in the first panel of Fig. 6.2. Computations have been performed on a Matlab suite, by using a laptop with CPU Intel Core i7-6700HQ at 2.60 GHz.

7 | Conclusions and Future Work

This thesis focuses mainly on the analysis and control of symbolic systems through regular language specifications. Due to the hugeness of this area, the problems and scenarios encountered are extremely different from each other. In order to better explain the different approaches adopted we list hereafter the results achieved in this thesis and their possible evolutions:

In **Chapter 4** we addressed symbolic control design of incrementally stable nonlinear systems with dynamic regular language specifications. An analysis of time computational complexity shows the benefits of the approach we proposed here over traditional ones. A possible future work could investigate how a change in the environment external to the plant can be translated into a dynamic specification. It could also be possible to extend the class of system considered, stochastic systems instead of incrementally stable nonlinear systems.

In **Chapter 5** we investigated the problem of enforcing reachability and safe reachability specifications on nondeterministic finite state systems through controllers combining feedforward and output feedback schemes. Necessary and sufficient conditions for the control problem to admit a solution were derived and a controller was designed. The strength of this approach lies in the generality of the systems considered, nondeterministic systems with a non-injective output function, and in the necessary and sufficient conditions for the existence of the controller.
A possible future direction of the results proposed in this chapter may regard the feedback scheme considered: we supposed that given a sequence of inputs we will receive an ordered sequence of outputs according with the

evolution of the system. Unordered or missing output sequences of some outputs are worth considering. Another possible extension may regard the class of systems considered, one may replace the nondeterministic model with a stochastic one. Last but not least a future direction is to incorporate further logical specifications.

In **Chapter 6** we addressed data–driven control design of an abstract system with specifications expressed in terms of regular languages and applied the results to the artificial pancreas. One future research direction concerns the extension of our results from data–driven control design of input–state abstract systems to input–state–output abstract systems. Also, it would be interesting to extend our results to specifications expressed in terms of infinity-time horizon properties as for example those expressed as Linear Temporal Logic.

Bibliography

Angeli, D. (2002). A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, *47*(3), 410-421.

Belta, C., Yordanov, B., & Aydin Gol, E. (2017). *Formal methods for discrete-time dynamical systems*. Springer.

Bogomolov, S., & Jungers, R. (2021). *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC).* New York, NY, USA: Association for Computing Machinery (ACM).

Borri, A., Cacace, F., De Gaetano, A., Germani, A., Manes, C., Palumbo, P., ... Pepe, P. (2017). Luenberger-like observers for nonlinear time-delay systems with application to the artificial pancreas: the attainment of good performance. *IEEE Control Systems*, *37*(4), 33–49.

Borri, A., Palumbo, P., Manes, C., Panunzi, S., & De Gaetano, A. (2017). Sampled-data observer-based glucose control for the artificial pancreas. *Acta Polytechnica Hungarica*, *14*(1), 79–94.

Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Kluwer Academic Publishers.

Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, p. 3387-3395).

Clarke, E. M., Grumberg, O., & Peled, D. (1999). *Model checking*. MIT Press.

Coulson, J., Lygeros, J., & Dorfler, F. (2019). Data-enabled predictive control: In the shallows of the DeePC. In *17th European Control Conference (ECC)* (pp. 25–28). Naples, Italy.

Dutta, S., Kushner, T., & Sankaranarayanan, S. (2018). Robust data-driven con-

trol of artificial pancreas systems using neural networks. In *Computational Methods in Systems Biology* (pp. 183–202). Springer.

Ghosh, S., Bansal, S., Sangiovanni-Vincentelli, A., Seshia, A., & Tomlin, C. (2019). A new simulation metric to determine safe environments and controllers for systems with unknown dynamics. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control* (pp. 185–196).

Girard, A., & Pappas, G. J. (2011). Approximate bisimulation: a bridge between computer science and control theory. *European Journal of Control*, *17*(5–6), 568–578.

Guo, Y. (2018). Observability of boolean control networks using parallel extension and set reachability. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(12), 6402–6408.

Hashimoto, K., Saoud, A., Kishida, M., Ushio, T., & Dimarogonas, D. (2020). Learning-based safe symbolic abstractions for nonlinear control systems. *arXiv preprint arXiv:2004.01879*.

Hou, Z., & Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, *235*, 3–35.

Kazemi, M., & Soudjani, S. (2020). Formal policy synthesis for continuous-space systems via reinforcement learning. *arXiv preprint arXiv:2005.01319*.

Kazumune, H., & Toshimitsu, U. (2020). A learning-based approach towards symbolic safety controller synthesis. In *Poster at HSCC conference.*

Khalil, H. K. (1996). *Nonlinear systems* (Second ed.). New Jersey: Prentice Hall.

Kushner, T., Bortz, D., Maahs, D. M., & Sankaranarayanan, S. (2018). A data-driven approach to artificial pancreas verification and synthesis. In *ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)* (pp. 242–252).

Lavaei, A., Somenzi, F., Soudjani, S., Trivedi, A., & Zamani, M. (2020). Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning. In *ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)* (p. 98-107).

Liu, J., & Ozay, N. (2014). Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC)* (pp. 293–302).

Masciulli, T., & Pola, G. (2020). On symbolic control design of nonlinear systems with dynamic regular language specifications. *IFAC-PapersOnLine*, *53*(2).

Masciulli, T., & Pola, G. (2021). Symbolic control design of incrementally stable nonlinear systems with dynamic regular language specifications. *Automatica*, *130*.

Masciulli, T., Pola, G., De Santis, E., & Di Benedetto, M. D. (2021). Output feedback reachability of controlled–observable states for nondeterministic finite–state systems. *IEEE Control Systems Letters*, *6*.

Pola, G., & Di Benedetto, M. D. (2019). Control of cyber-physical-systems with logic specifications: A formal methods approach. *Annual Reviews in Control*, *47*, 178–192.

Pola, G., Di Benedetto, M. D., & Borri, A. (2019). Symbolic control design of nonlinear systems with outputs. *Automatica*, *109*, 108511.

Pola, G., Masciulli, T., De Santis, E., & Di Benedetto, M. D. (2020). On data-driven controller synthesis with regular language specifications. *IFAC-PapersOnLine*, *53*(2).

Pola, G., Masciulli, T., De Santis, E., & Di Benedetto, M. D. (2021). Data-driven controller synthesis for abstract systems with regular language specifications. *Automatica*, *134*.

Pola, G., Pepe, P., & Di Benedetto, M. D. (2018). Decentralized approximate supervisory control of networks of nonlinear control systems. *IEEE Transactions on Automatic Control*, *63*, 2803–2817.

Ruberti, A., & Isidori, A. (1979). *Teoria dei sistemi*. Bollati Boringhieri.

Tabuada, P. (2008). An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, *53*(6), 1406-1418.

Tabuada, P. (2009). *Verification and control of hybrid systems: A symbolic approach*. Springer.

Wu, M., Yan, G., Lin, Z., & Lan, Y. (2009). Synthesis of output feedback control for motion planning based on LTL specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (p. 5071-5075).

Zhang, Z., Xia, C., & Chen, Z. (2020). On the stabilization of nondeterministic finite automata via static output feedback. *Applied Mathematics and Computation*, *365*, 124687.