Analysis of Hierarchical Structures for Hybrid Control Systems

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(**Dr.-Ing.**)
genehmigte Dissertation

vorgelegt von
Dipl.-Ing. Dmitry Gromov
geboren am 09.12.1974 in Minsk, Belarus

Promotionsausschuss:

Vorsitzender:	Prof. Dr. O. Brock
Berichter:	Prof. Dr.-Ing. J. Raisch
Berichter:	Prof. Dr.-Ing. T. Moor

Tag der wissenschaftlichen Aussprache: 27.08.2010

Berlin, 2010

D 83

to my parents

Acknowledgments

I would like to thank Professor Jörg Raisch for his help and his excellent supervision. His enthusiasm and constant encouragement have been a true inspiration. This project has been a great challenge and a wonderful learning experience.

Many thanks go out to Professor Alessandro Giua, Professor Thomas Moor, Stephanie Geist, Daniele Corona and Carla Seatzu for the many fruitful discussions that shaped this thesis.

I would also like to thank all the former colleagues from the Max-Planck Institut Magdeburg, Otto-von-Guericke University of Magdeburg and Technical University of Berlin for the great time I had with them, on many occasions, during the duration of my PhD.

Finally, I want to thank my family and my wife Ekaterina for their love and support during all these years.

Contents

1 Introduction and Literature Survey **1**
- 1.1 Hierarchical systems theory . . . 3
  - 1.1.1 Multilevel hierarchical control . . . 3
  - 1.1.2 Multilayer hierarchical control . . . 5
  - 1.1.3 Model approximation and abstraction . . . 6
- 1.2 Hybrid systems . . . 13

2 Ingredients from Behavioural Systems Theory **14**
- 2.1 Dynamical systems, behaviours and their properties . . . 14
- 2.2 Systems with inputs and outputs . . . 20
- 2.3 State systems . . . 22
  - 2.3.1 Dynamical systems in state space form . . . 22
  - 2.3.2 I/S/- machines . . . 25
- 2.4 Interconnection of dynamical systems . . . 28
  - 2.4.1 Behavioural description . . . 29
  - 2.4.2 Interconnection of state space dynamical systems . . . 32
- 2.5 Supervisory control problem . . . 35

3 Hierarchical Control **37**
- 3.1 Two-level hierarchical control architecture . . . 38
- 3.2 Bottom-up design strategy . . . 41
  - 3.2.1 Modelling issues . . . 44
  - 3.2.2 Qualitative behaviour shaping . . . 44
- 3.3 Non-conflictingness conditions . . . 47
- 3.4 Interface layer . . . 51
  - 3.4.1 Interface layer: functioning . . . 52
  - 3.4.2 Behavioural description . . . 54
  - 3.4.3 Non-conflictingness conditions . . . 56
- 3.5 Reconfigurable controller . . . 58
  - 3.5.1 Multi-controller . . . 58
  - 3.5.2 Reconfigurable controller with state sharing . . . 60
- 3.6 Multi-level hierarchy: non-conflictingness . . . 61

4 Optimal Control of Hybrid Automata under Safety and Liveness Constraints 65
4.1 Plant Model and Specifications 67
4.1.1 Hybrid Automata 67
4.1.2 The Plant Model 69
4.1.3 Specifications and problem decomposition 69
4.2 The low-level task 70
4.2.1 Ordered set of discrete abstractions 70
4.2.2 Specifications and supervisor design 74
4.2.3 Low-level controller 77
4.2.4 Plant model under low-level control 77
4.3 The high-level task 78
4.3.1 The optimal control problem with a finite number of switches 79
4.3.2 The optimal control problem with an infinite number of switches 84
4.3.3 The optimal control of switched systems with unstable dynamics 85
4.3.4 Robustness of the procedure 87
4.4 Numerical example 88

5 Detecting and Enforcing Monotonicity for Hybrid Control Systems Synthesis 92
5.1 Partial order relations 93
5.2 Monotone dynamical systems 94
5.2.1 Autonomous systems 94
5.2.2 Controlled systems 96
5.2.3 Special cases 98
5.3 The role of monotonicity in abstraction based control synthesis 99
5.4 Monotonisation through feedback 100
5.4.1 Behavioural description 100
5.4.2 Low-level controller design 101
5.4.3 Example 102

6 Conclusion 105

List of Figures

1.1 A multilevel hierarchical control system 4
1.2 A multilayer hierarchical control system 5
1.3 Reduced model, over- and under-approximation 8
1.4 Model aggregation 8
1.5 A (feedback-based) abstraction 10
1.6 A partition machine 12

2.1 Electrical circuits 16
2.2 Backward shift 17
2.3 Electro-magnetic circuit 22
2.4 Moore and Mealy automata 28
2.5 Interconnection of two dynamical systems 29
2.6 Feedback interconnection 30
2.7 Two Mealy automata 31

3.1 Hierarchical control architecture 39
3.2 An event generator 52
3.3 A partition and a cover of the set Y_L 54
3.4 Reconfigurable controller with jumps in the state 58
3.5 Reconfigurable controller with continuous state evolution . . 60
3.6 Multi-level hierarchy 62

4.1 A hierarchical control architecture 66
4.2 A graph describing an hybrid automaton. 68
4.3 Moore-automaton and an equivalent automaton without outputs 73
4.4 Graph of the automaton $\mathcal{A}$ (continuous) and $\overline{\mathcal{A}}$ (continuous and dashed) described in the example. 86
4.5 Discrete time trajectories of dynamics A_1 and A_2, with eigenvalues along the unitary circle. 89
4.6 Invariants of locations 1 and 2 and the forbidden region $X_f = X \setminus (inv_1 \bigcup inv_2)$ defined in Def. 4.3.3. 90
4.7 Switching table of the problem $\overline{OP}(\overline{\mathcal{A}})$ defined in the example. 90

4.8 Trajectories $x(k)$ and $i(k)$ of the optimal solution of $OP(\mathcal{A})$ for an initial point $x_0 = [-1\ 0]'$ with $i_0 = 1$ 91

Chapter 1

Introduction and Literature Survey

Hybrid systems arise from nontrivial interaction between discrete-event subsystems (usually modelled by finite automata) and continuous subsystems (described by differential or difference equations). Therefore, neither methods from discrete-event systems theory nor methods from continuous systems theory alone can be used to adequately analyse and control systems of this kind. An additional difficulty is that the restrictions imposed on the system often refer to different aspects of the system dynamics and, hence, cannot be addressed in a uniform way. There are two widely used approaches aimed at overcoming this problem. The first one, called *continuation*, is based on the conversion of a hybrid system into a purely continuous dynamical system (see, e.g., [14, 13]). This approach has several drawbacks, among which is increased complexity of the resulting system dynamics (see [12] for a detailed analysis). The second approach is to approximate continuous dynamics by a discrete event system (see, e.g., [27, 58, 21, 65]). In this way, the hybrid control problem is transformed into a purely discrete one. In particular, in [78], Moor and Raisch proposed an approximation-based approach set within Willems' behavioural system theory framework. This approach allows to approximate a hybrid system by a nondeterministic finite automaton which can be effectively treated by the Ramadge-Wonham supervisory control theory. As many other approximation-based approaches, it also suffers from the "curse of complexity". The state set of the approximating automata can grow enormously even for comparatively simple systems. Hence, complexity becomes a key problem in control design for hybrid systems.

A hierarchical control approach could be a way out of this problem. Obviously, a decomposition of the controller into several levels can signif-

icantly facilitate the design procedure. Moreover, it allows an engineer to introduce some intuition into the control systems design. A theoretical basis for the hierarchical control synthesis was developed within the behavioural systems theory framework (see [97] and references therein). Unfortunately, there are still only few practical applications of this approach. Moreover, the conditions proposed in [97] are not always easy to check in practice.

The dissertation is intended to close this gap between theory and practice. Its main contribution consists in the following:

- A unified framework which allows for a constructive description and analysis of all levels of a hierarchical control structure is proposed. This framework is based on realising control systems by state machines. We show that most practically relevant classes of control systems can be described as input/output state machines and give a classification of such state machines.

- Developing the results obtained in [97], we formulate a set of constructively verifiable conditions which guarantee a non-conflicting interaction of all control levels.

- Two particularly useful classes of intermediate control layers are described and analysed with respect to the non-conflictingness property.

- We consider two cases where the developed approach is applied to practically relevant control problems. In the first case, we consider an optimisation problem for a hybrid system under safety and liveness conditions. It is shown that this problem can be efficiently solved in a hierarchic way: the low-level controller enforces safety and liveness constraints while the high-level controller performs optimisation. In the second case, the low-level controller is designed to render the plant monotone.

The thesis is arranged as follows: in this chapter, we give a review of different approaches to hierarchical control systems design and introduce some concepts which will be extensively used in the sequel. Chapter 2 provides all necessary information from behavioural systems theory as well as the notion of supervisory control. In Chapter 2, we also consider the class of I/S/-machines and analyse their properties. Furthermore, for the case of feedback interconnection of two dynamical systems we formulate two theorems which will serve as the foundation for the subsequent results. In Chapter 3, we consider both two-level and multi-level hierarchical control architectures. Moreover, we define non-conflictingness conditions and analyse two particularly important types of intermediate control layer. Chapter 4 is devoted to

optimal control of a discrete-time hybrid automaton under safety and liveness constraints. Finally, in Chapter 5 we show how a hierarchical approach can be used to simplify the system dynamics. It contains new results on how to efficiently check and enforce monotonicity.

1.1 Hierarchical systems theory

The use of hierarchical concepts for the synthesis of control systems can be traced back to the mid 50's, when first results on cascade control systems were published [35]. Cascade control can be seen as a precursor of hierarchical control. Indeed, while designing the outer feedback loops in a control cascade, one assumes that the behaviour of the inner closed-loop subsystem can be described by a fairly simple model. These ideas were further developed in [32], where the basic principles of the hierarchical control architecture were formulated (albeit not referring to the hierarchy as such). Later on, during the 60's, a vast body of theoretical research on the foundations of hierarchical systems theory was done, which culminated in the seminal book "Theory of Hierarchical, Multilevel Systems" [77], by Mesarović, Macko, and Takahara, where the basic principles of *multilevel* hierarchical control have been formulated. Simultaneously, the theory of *multilayer* hierarchical control was developed. The difference between these two theories can be expressed in the following keywords: *"decomposition"* and *"coordination"* for multilevel systems theory, and *"aggregation"* and *"multiple time-scales"* for its multilayer counterpart. In this work, we develop an approach which is based on the ideas from both multilevel and multilayer hierarchical control theories. Below, we give a short description of these theories.

1.1.1 Multilevel hierarchical control

In multilevel control theory, the control system is assumed to be decomposable in a set of coupled subsystems. The decomposition can be done based on the physical or logical system structure. Each subsystem is connected to a local controller which solves a particular local control problem. In order to achieve the overall control goal and to ensure consistent functioning of all subsystems, a higher level controller is designed to coordinate the local controllers by modifying their individual goals. In a two-level hierarchical control system, shown in Fig. 1.1, the high-level controller (coordinator) modifies the local goals by setting parameters γ_i, which are referred to as coordination parameters. The high-level feedback variables ω_i contain information about the performance of the local controllers C_i.

In [77], the basic notions of *coordinability* and *consistency* were defined. A system is said to be coordinable if there exists a supremal coordination

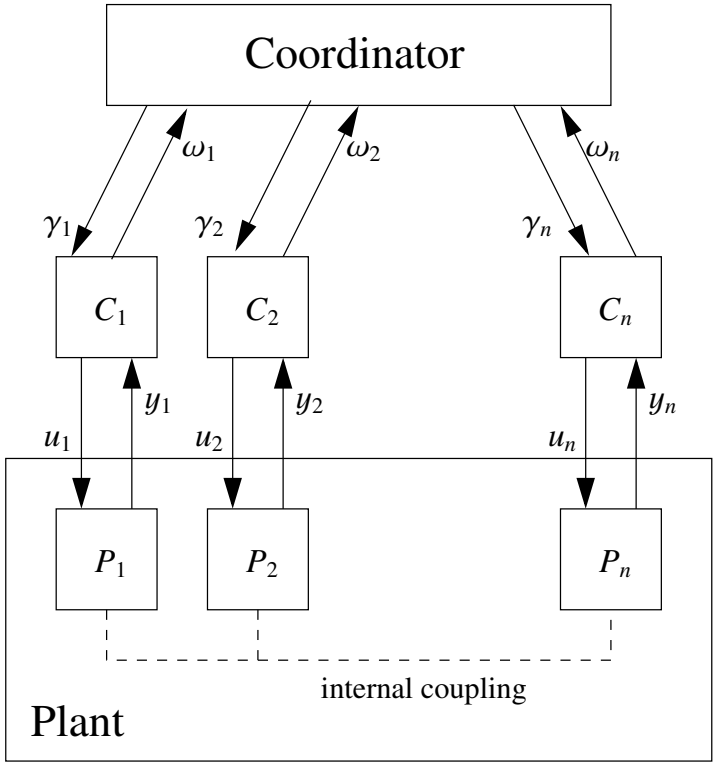

Figure 1.1: A multilevel hierarchical control system

control (i.e., values of coordination parameters) such that a solution of all local control problems exists. The consistency postulate says that whenever the high-level and local controllers can solve their respective problems, an overall solution exists. Furthermore, the notions of feasible and unfeasible problem decompositions were introduced. The latter case refers to the fact that while the coordinating parameters are determined using an iterative optimisation procedure, only the final solution respects constraints imposed on the system. For more details about these notions see [73] and references therein.

The theoretical concepts developed by Mesarović and his coworkers were put into a concrete dynamical systems framework by Singh [108], Findeisen *et al.* [34], and Stoilov and Stoilova [112]. It is worth noting that multilevel control ideas are mainly applied to the solution of complex optimisation problems, both static and dynamic, where the overall optimisation problem can be decomposed in a set of coupled subproblems which can be treated (relatively) independently. The coordinator optimises an overall cost function while respecting restrictions imposed on the whole system.

1.1.2 Multilayer hierarchical control

In contrast to multilevel structures, multilayer hierarchical control theory is primarily concerned with different representations (models) of a complex plant. In a multilayer structure the specification of control is split into algorithms which operate on different time scales and take their decisions based on models of different granularity. Note also that the high-"level" models must take into account the low-"level" control algorithms, i.e., represent the plant under low-"level" control. A symbolic representation of a multilayer structure is shown in Fig. 1.2.

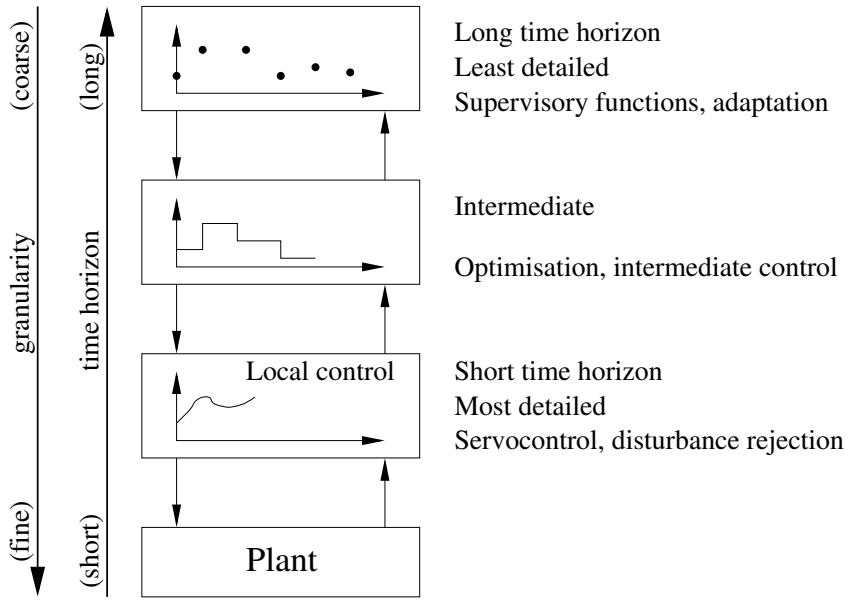

Figure 1.2: A multilayer hierarchical control system

Below, we briefly characterise the typical features of a multilayer hierarchical structure [34, 66, 109, 115]:

1. **Functional hierarchies.** A specific feature of a multilayer control architecture is that different layers have a different functionality. Although the allocation of tasks between different layers strongly depends on the particular problem, we can formulate some general principles. The controllers at the bottom of the hierarchy are designed to keep the system at a prescribed operating point (regime). These local controllers are designed to cope with disturbances, thereby allowing higher layers to use a simplified system description when defining their control objectives. The tasks of the intermediate layers may

include, e.g., optimisation and setting the operation points for the servocontrollers. The top layer, in turn, is responsible for the long term functioning of the system. It modifies the system structure to adapt to changing external conditions. In manufacturing applications, the highest layer is often in charge of the long term scheduling.

2. **Time-scale decomposition.** Different problems solved at the respective layers require different time scales. The local controllers work either in continuous time or in discrete time with rather high sampling rates. They have to react immediately while having information over a small time interval. The time window of intermediate controllers is usually much larger. The highest layer, in turn, evolves in logical time formed by a sequence of events produced by the lower layers.

3. **Hierarchy of models.** Each of the layers has a different model of the controlled system; the model used at the top layer is least detailed. At the same time, the models used by the higher layers must incorporate the lower layers. The procedure of building appropriate models of the process is called model approximation or model abstraction. These issues are described in detail in Sec. 1.1.3.

The multilayer hierarchical control architecture has been used in a wide variety of applications: intelligent control for unmanned aerial vehicles [11], process control [28], robotic control [16], and water management [93], just to mention a few. In [39], a hierarchical structure of the Intelligent Vehicle Highway System (IVHS) was described. The proposed hierarchical control architecture consists of the following layers: the network layer (at the top), the link layer, the coordination layer, the regulation layer, and the physical layer (plant).

In [62], Larsson and Skogestad consider a typical process control system as a hierarchical structure with several layers. The layers are distinguished according to their time-scales: scheduling (weeks), site-wide optimisation (days), local optimisation (hours), supervisory/predictive control (minutes) and regulatory control (seconds). It can be easily seen that these layers do also differ with respect to the amount (and character) of information they need for the fulfilment of their tasks.

1.1.3 Model approximation and abstraction

In this section, we discuss different approaches to a simplified description of complex systems. This is a central problem in hierarchical control design since each particular control layer requires a specific plant model. These models can be more or less specialised depending on the control problem

solved by the layer. Below, we consider several concepts related to this problem.

Approximation refers to a procedure which replaces a complex system Σ with its simplified model Σ_{app}, while retaining the main features of the original system.

One particular case of approximation is called *model reduction*. In this case we say that the system Σ is replaced by its reduced model Σ_r. There are a number of features which (completely or partially) may characterise the reduced model Σ_r:

- The dynamics of Σ_r is sufficiently close to the dynamics of the original problem. To estimate the closeness of both dynamics an approximation error is calculated. The global error bound gives a quantitative estimation for the precision of the approximation.
- Stability, controllability and further qualitative characteristics are preserved.
- The reduced model can be treated analytically in an easier way than the original system.

There is a vast body of research devoted to the approximation/model reduction of complex systems (see, e.g., [87, 3, 9]). Various methods exist which allow to find a reduced model with respect to certain system characteristics with a prescribed accuracy. The dynamics of the model Σ_r can be kept rather close to the dynamics of the initial system. However, in many safety-critical applications the designer wants to ensure that the approximation does indeed contain the whole set of trajectories of the initial system. On the other hand, in many cases the approximation may not contain any signals which are not compatible with the original system. To capture these requirements the notions of *over-approximation* and *under-approximation* have been introduced. In general, Σ_o is said to be an over-approximation of Σ if any trajectory σ compatible with Σ is also compatible with Σ_o. An over-approximation is often referred to as a *conservative approximation*. On the other hand, the system Σ_u is said to be an under-approximation of Σ if any trajectory σ compatible with Σ_u is also compatible with Σ. Fig. 1.3 gives a schematic illustration of approximation, over- and under-approximation.

One particularly interesting approach is based on selecting some specific properties of the system dynamics and analysing the system behaviour with respect to these properties while neglecting remaining aspects. This procedure is called *aggregation*. The idea of *aggregation* is intimately connected with the notion of *equivalence relation* [10]. An equivalence relation is a binary relation $a \sim b$ defined for any two elements of some set X. An equivalence relation is reflexive, symmetric and transitive. The *equivalence class*

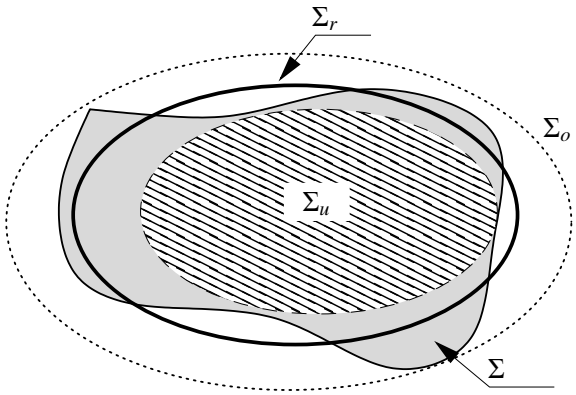

Figure 1.3: Reduced model, over- and under-approximation

of a under $\sim$, denoted $[a]$, is defined as

$$[a] = \{b \in X | a \sim b\}.$$

The set of all equivalence classes of X, denoted $X/\sim$, is referred to as the *quotient set* of X by $\sim$.

Let $\Psi : X \to Y$ be a surjective mapping from X onto Y. This mapping also introduces an equivalence relation. We say that x' and x'' from X are equivalent, i.e., $x' \sim x''$, if $\Psi(x') = \Psi(x'')$. Thus, one can define the quotient set of X by Ψ, denoted X/Ψ. We say that the mapping Ψ aggregates the set X by grouping the elements from one equivalence class.

Aggregation can also be applied to dynamical systems. Fig. 1.4 shows a dynamical system Σ with input signal $u \in \mathcal{U}$ and output signal $y(t) = \Psi(x(t))$, where $x(t)$ is the state at time instant t. A dynamical system Σ_{agg} is said to be an *aggregated model* of Σ if it possesses the same I/O dynamics as Σ.

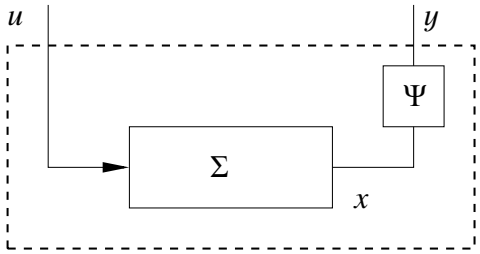

Figure 1.4: Model aggregation

We illustrate the notion of model aggregation for the case of a linear time-invariant dynamical system [4, 60]

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{1.1}$$

where $x(t) \in X$, $u(t) \in U$ and X and U are vector spaces. Let S be a linear subspace of X. It introduces an equivalence relation in the following way: $x', x'' \in X$ are said to be equivalent if $x'' - x' \in S$. The quotient set X/S can be defined in a usual way. Moreover, we can define a vector addition and scalar multiplication on X/S which turns X/S into a vector space. Thus, the aggregation mapping Ψ reduces to a linear transformation $C : X \to X/S$:

$$y(t) = Cx(t). \tag{1.2}$$

We want to represent the aggregated model of (1.1),(1.2) as a linear system

$$\dot{y}(t) = Fy(t) + Gu(t). \tag{1.3}$$

The relationship between (1.1) and (1.3) can be described by the following diagram:

$$\begin{array}{ccccc} & & X & \xrightarrow{A} & X \\ & \overset{B}{\nearrow} & & & \\ U & & \downarrow C & & \downarrow C \\ & \underset{G}{\searrow} & & & \\ & & X/S & \xrightarrow{F} & X/S \end{array} \tag{1.4}$$

We say that aggregation is perfect if the diagram (1.4) commutes, i.e., if $G = CB$ and $FC = CA$. Moreover, the initial states have to be consistent: $y(0) = Cx(0)$. Later, Kwong and Zheng [61] extended this approach to nonlinear systems, but the obtained conditions for the existence of an aggregated model turned out to be rather restrictive.

Recently, the concept of *abstraction* has drawn particular attention of (hybrid) control theorists [90, 1, 114]. The abstraction of the system is defined to be another system which preserves certain properties of the original system while ignoring details. As a rule, the abstracted system has a state space of less cardinality than that of the original system or is structurally simpler (see Fig. 1.5 for illustration). The main advantage of using such simplified models is that one can use well developed formal methods for analysis and design of control mechanisms for complex systems [1, 46].

The abstraction procedure is so closely related to approximation and aggregation that these terms are often used synonymously. At the same time, there are some characteristic features for abstractions. Typically, the input and output sets of an abstracted system have less cardinality than those of the original one. This distinguishes abstraction from aggregation, since an aggregated system normally has the same control inputs. Thus, one has to establish the correspondence between the trajectories of the abstracted and original system. This is done by an *interface layer* Σ_{int} whose functioning is described in detail in Sec. 3.4. Furthermore, since the abstracted system

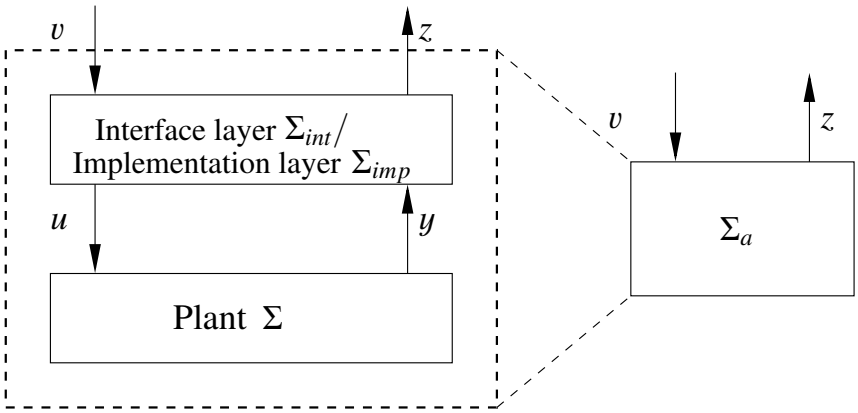

Figure 1.5: A (feedback-based) abstraction

has a simpler structure/dynamics than the underlying complex system, its I/O behaviour does not coincide with the I/O behaviour of the composite system $\Sigma_{int}[\Sigma]$. An exact correspondence can be achieved only for discrete-event systems or for systems with strongly restricted continuous dynamics [118, 113]. This is referred to as *bisimulation*. For more complex dynamical systems, one requires the I/O behaviour of the abstracted system to over-approximate that of $\Sigma_{int}[\Sigma]$. It is said that the abstraction has to be *conservative*.

There exist different abstraction techniques which can be classified according to the class of the original/abstracted system. For instance, Asarin et al. [6] propose to abstract complex dynamical systems by hybrid automata with simpler continuous dynamics. In [94, 30], abstractions of hybrid automata by timed automata were studied. One of the most widely used abstraction technique is the abstraction of complex (hybrid) systems by discrete-event systems, in particular by finite automata. The choice of finite automata is due to the fact that the theory of discrete-event systems provides a wide range of well-developed and computationally efficient methods for analysis, verification, and supervisory control design [20, 124].

During the last decade, there have been a number of results in this direction. In [27], continuous hybrid systems are over-approximated by a special class of finite automata, namely Muller automata. The approximation procedure is based on the subsequent computation of forward and backward reachable sets. In [22], Chutinan and Krogh proposed a method to compute discrete abstraction for polyhedral invariant hybrid automata. State transitions of the abstracted system are determined by computing polyhedral approximations to the reachable sets. Lunze and Schröder [69, 103] used stochastic automata to approximate hybrid and continuous systems.

As mentioned above, discrete abstractions have to provide an over-approximation of system's behaviour. Therefore, it may happen that a solution of a control problem cannot be found since this over-approximation may be too coarse. To overcome this difficulty, the notion of ℓ-complete abstractions has been proposed [98, 96]. This approach allows for computing a set of abstractions ordered in the sense of approximation accuracy. Later, this idea was extended by an iterative procedure alternating abstraction refinement with trial controller synthesis [83].

One particularly fruitful approach, which has proved to be very useful for different classes of systems is a *feedback-based abstraction*. In this approach, the system Σ with input and output signals (u, y), is abstracted by a (simplified) model Σ_a, whose input and output (v, z) are not identical to the I/O signals of the original system. Furthermore, an implementation layer Σ_{imp}, enforcing the system to match the abstraction Σ_a is introduced. This is illustrated in Fig. 1.5. The function of Σ_{imp} is twofold. First, it performs the aggregation of the low-level signal: $z = \Psi(y)$ (in some cases Ψ may be chosen to be the identity). Second, it chooses the low-level control signal u in such a way that the composite system $\Sigma_{imp}[\Sigma]$ is externally equivalent to the abstraction Σ_a. The control signal u is chosen as a function of both the high-level input signal v and the low-level output y.

Below, we consider a couple of feedback-based abstraction techniques.

In [125], Zhong and Wonham proposed a hierarchical structure for the supervisory control of discrete-event systems. The concept of *hierarchical consistency* is proposed, which means that "the model of control available at any given level of hierarchy can be utilised with assurance that the next level down will respond as required or expected" [125]. Later, an abstraction procedure, based on the state space aggregation, was proposed by Caines and Wei [18] for finite automata and further extended to the case of continuous systems [19]. This approach is based on the decomposition of the state space of the system in a set of disjoined subsets (partition blocks): $\pi = \{X_i\}_{i=1,|\pi|}$. A pair (X_i, X_j) of partition elements[1] is said to dynamically consistent if for any $x' \in X_i$ there exists a low-level (feedback) control u, which steers the state into X_j without visiting other blocks. Then, the initial system Σ can be abstracted by a partition machine Σ_{pm}, where the high-level transitions are defined for all dynamically consistent pairs (X_i, X_j). This structure is illustrated in Fig. 1.6.

Further results on hybrid partition machines can be found in [64, 65]. It should be noted, however, that the determination of a dynamically consistent partition is a rather involved problem which can be solved successfully only for particular classes of control problems.

[1] where the indices may be equal, $i = j$.

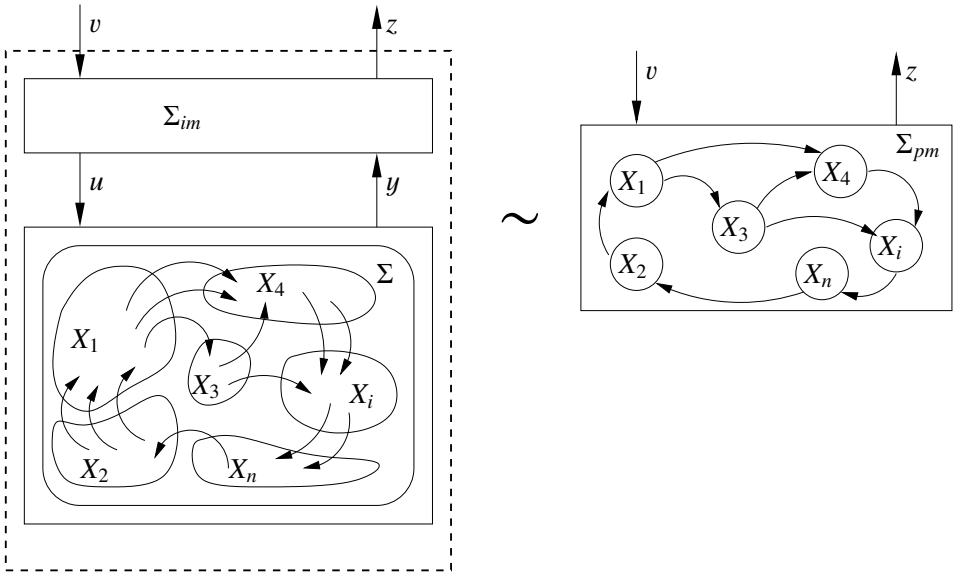

Figure 1.6: A partition machine

Using methods from differential geometry, Pappas *et al.* developed the theory of continuous abstractions of dynamical and control systems [90, 92, 91] (see also [76] for further details and examples). They introduced the notion of Φ-related control systems (compare, e.g., with the notion of F-related vector fields in [63, Ch. 4]) to establish a relation between the original control system Σ and its continuous abstraction Σ_{ca}. A distinctive feature of this approach is that it establishes a relation between the vector fields of both systems. Therefore, the system Σ_{ca} cannot serve as an abstraction in the proper sense since there is no rule which would establish a direct connection between the trajectories of Σ_{ca} and Σ. There can even be problems related to well-posedness issues as was mentioned in [41]. However, this sort of abstractions can be used for the reasoning about qualitative properties of the underlying complex system such as, e.g., controllability [92].

Recently, Girard and Pappas [38] have proposed to use an interface which forms the low-level control u in such a way that the original system Σ traces Σ_{ca} within some computable bounds. This puts the continuous approximations approach in the context of feedback-based approximation.

An example of feedback-based abstraction methods can be found in [42], where the intermediate level was designed to make the underlying system monotone. Then, the abstraction of the system behaviour can be easily calculated while guaranteeing the conservativeness property. This will be investigated in more detail in Ch. 5. Similar ideas were also used in [44, 37].

1.2 Hybrid systems

Since the late 80's, hybrid dynamic systems have become a very "fashionable" topic of research within the control engineering community. The main reason for this lies in the particular flexibility and expressiveness of hybrid systems. Actually, there is rather a family of hybrid modelling concepts than a unique one. However, there is one specific feature which is common for all hybrid systems. They are characterised by a non-trivial interaction between continuous and discrete dynamics. Because of this, hybrid dynamic systems are very useful for modelling of complex technical systems which combine both continuously operated plants and logic/discrete controllers. Moreover, hybrid modelling formalism proved to be very useful for modelling these systems at different levels of abstraction.

Here, we do not intend to give an extensive overview of the topic. A good introduction to hybrid systems can be found in the lecture notes [104]. For a detailed description of the semantics and dynamics of hybrid systems the reader is referred to a number of proceedings volumes, e.g., [15, 33, 74] and to recent papers [40, 8, 51, 52, 70]. Very recently, a handbook of hybrid systems control has been published [68].

There are many special classes of hybrid systems which differ in the type of governing dynamical equations, in the structure of discrete transitions and so on. These are piece-wise affine systems (PWA), switched systems, hybrid automata (HA), and impulsive hybrid systems – just to mention a few. Moreover, a hybrid system may communicate with its environment in two different ways: synchronously and asynchronously. This important problem is studied in details in Chapter 3, in particular in Sec. 3.4. Later on, we will give a detailed description of particular classes of hybrid systems directly in those sections where we will use them.

Chapter 2

Ingredients from Behavioural Systems Theory

Behavioural systems theory has been pioneered by Jan Willems in a series of papers, e.g., [121, 122]. The main idea was to describe the behaviour of a dynamical system as a set of all signals which the system can potentially produce (or, in other words, which are compatible with the system). Behavioural systems theory allows one to study dynamical systems of different nature (e.g., discrete-event, continuous- or discrete-time) in a uniform way and, hence, can be used for the investigation of generic properties of these systems.

In Section 2.1, we recall some basic concepts, which will be intensively used in the sequel. Further, in Section 2.2, we give a definition of a special class of dynamical systems, namely I/- systems (firstly introduced in [78]). These systems can be seen as a generalisation of conventional input/output systems. Section 2.3 is devoted to dynamical systems with a state space structure. These systems can be constructively described as state machines. In the following, we will widely use both behavioural and state space descriptions of dynamical systems. In Section 2.4, we consider the interconnection of two dynamical systems and study its properties. Finally, in Section 2.5, we formulate a supervisory control problem within the behavioural framework.

2.1 Dynamical systems, behaviours and their properties

In this section, we recall the notion of a dynamical system and its behaviour. Further, we define and discuss some properties of behaviours which will be used in the sequel. Most definitions are adopted from the papers [121, 122].

Definition 2.1.1 ([121]) *A dynamical system Σ is defined as a triple*

$$\Sigma = (\mathbb{T}, W, \mathcal{B}),$$

where $\mathbb{T}$ is the time axis, W is an abstract set, called the signal alphabet, and $\mathcal{B} \subseteq W^{\mathbb{T}}$ is the behaviour.

We will follow a constructive approach and assume that system's behaviour consists of all signals that satisfy a certain set of laws. These laws can be described by difference, differential or algebraic equations (or inequalities) and their combinations. These equations may also contain Boolean expressions. We call these equations *behavioural equations*. Furthermore, from now on we will use the terms "dynamical system" and "behaviour" interchangeably. Obviously, a signal is compatible with the dynamical system if it belongs to the system behaviour and vice versa. The statement that a system possesses some property implies that its behaviour has this property too.

To specify the class of problems we are dealing with we make several assumptions:

A1. The time axis $\mathbb{T}$ is a linearly ordered additive semi-group (i.e., $\{t_1, t_2 \in \mathbb{T}\} \Rightarrow \{t_1 + t_2 \in \mathbb{T}\}$) with a minimal element (i.e., $\exists \tau \in \mathbb{T}$ s.t. $\tau \leq t \; \forall t \in \mathbb{T}$). The order relation $\leq$ is introduced in a natural way.

A2. The behavioural (dynamical) equations of the processes under investigation are time-invariant, i.e., they do not change under the transformation $t' = t + \Delta t, \; \Delta t \in \mathbb{T}$.

Thus, there exists an initial time instant. This is a very natural assumption. Moreover, due to the second assumption, the initial time instant can be identified with zero. In the following, without loss of generality, we will consider only two cases: $\mathbb{T} = \mathbb{R}_{\geq 0}$ and $\mathbb{T} = \mathbb{N}_0$. A more general case can be easily addressed at the cost of increased notational complexity.

Let us now consider a simple example of electrical systems and their associated behaviours.

Example 2.1.2 *In Fig. 2.1a) a linear electrical circuit is shown. Let's assume that we are interested in voltage $v(t)$ and current $i(t)$, $t \in \mathbb{R}_{\geq 0}$. The behavioural equations are*

$$\begin{cases} v_R = R\, i_R & \text{(Ohm}'\text{s equation)} \\ v_L = L \frac{d}{dt} i_L & \text{(Inductance equation)} \\ i_L = i_R = i & \text{(Kirchhoff}'\text{s current law)} \\ v_L + v_R = v & \text{(Kirchhoff}'\text{s voltage law)} \end{cases}$$

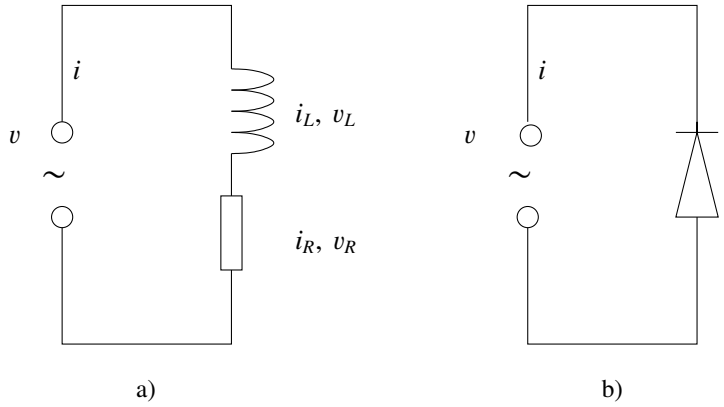

Figure 2.1: Electrical circuits

After elimination of variables, we obtain a behavioural equation with respect to the current i and the voltage v:

$$R\,i + L\frac{d}{dt}i = v.$$

Hence, the system behaviour is described as

$$\mathcal{B}_a = \{(i,v) \in (I \times V)^{\mathbb{T}} \mid Ri + L\frac{d}{dt}i = v\},$$

where $I = V = \mathbb{R}$.

In the case of a nonlinear circuit (see Fig. 2.1b)) the behaviour is specified by a combination of linear and Boolean equations:

$$\mathcal{B}_b = \{(i,v) \in (I \times V)^{\mathbb{T}} \mid (i = k_1 v)[v \geq 0] \vee (i = k_2 v)[v < 0]\}.$$

In the following, we consider several general properties of behaviours. As in the case of conventional dynamical systems, we begin with the property of time-invariance. This property can be easily defined in the framework of behavioural systems. Note that time-invariance of a behaviour is in general not equivalent to time-invariance of the system equations. We will address the question of the interrelation of these two properties later, in Sec. 2.3.

Definition 2.1.3 (Time invariance, [121]) *The dynamical system $\Sigma = (\mathbb{T}, W, \mathcal{B})$ is said to be time-invariant if $\sigma^t\mathcal{B} \subseteq \mathcal{B}$ for all $t \in \mathbb{T}$, where σ^t denotes the backward or left shift: $(\sigma^t f)(t') = f(t+t')$, $\sigma^t\mathcal{B} = \{\sigma^t w \,|\, w \in \mathcal{B}\}$ (see Fig. 2.2).*

One important question is the relation of the behaviour and its constituent signals. It turns out that in many cases it suffices to analyse only finite fragments of a signal to decide whether it belongs to a certain behaviour

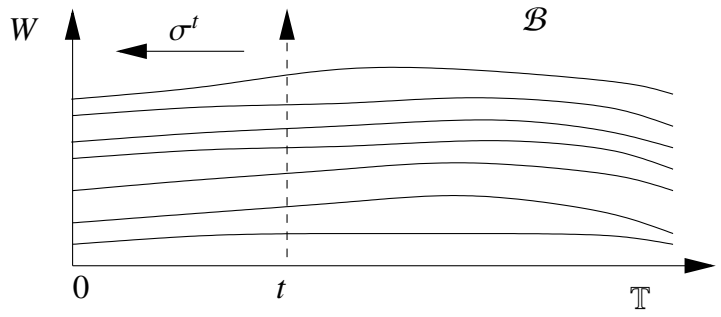

Figure 2.2: Backward shift

or not. Such behaviours are called *complete*. In practice, this means that the equations determining the system behaviour do not involve integration or summation over infinite intervals.

Definition 2.1.4 (Completeness, [121]) *The dynamical system* $\Sigma = (\mathbb{T}, W, \mathcal{B})$ *is said to be complete if*

$$(w \in \mathcal{B}) \Leftrightarrow \left(w|_{[t_1,t_2]} \in \mathcal{B}|_{[t_1,t_2]} \; \forall \; t_1, t_2 \in \mathbb{T}, \; t_1 \leq t_2\right),$$

where $w|_{[t_1,t_2]}$ *is the restriction of signal* w *on the interval* $[t_1, t_2]$.

The next definition says that for some systems it is sufficient to consider only fragments of fixed length ℓ. These systems are called ℓ*-complete*.

Definition 2.1.5 (ℓ-completeness, [121]) *The dynamical system* $\Sigma = (\mathbb{T}, W, \mathcal{B})$ *is said to be* ℓ*-complete if there exists* $\ell \in \mathbb{T}$ *such that*

$$(w \in \mathcal{B}) \Leftrightarrow \left(w|_{[t,t+\ell]} \in \mathcal{B}|_{[t,t+\ell]} \; \forall \; t \in \mathbb{T}\right).$$

If the system is 0-complete, it is called **instantly specified** *(or* **memoryless***).*

The obvious conclusion is that any ℓ-complete system is complete. In certain cases the property of completeness can be proved formally. The following lemma gives necessary and sufficient conditions for a linear discrete time system to be complete.

Lemma 2.1.6 ([120], Prop. 4) *Consider* $\Sigma = (\mathbb{T}, \mathbb{R}^q, \mathcal{B})$ *with* $\mathbb{T} = \mathbb{N}_0$. *The system* Σ *is linear and complete iff* $\mathcal{B}$ *is a linear subspace of* $(\mathbb{R}^q)^{\mathbb{T}}$ *which is closed in the topology of pointwise convergence.*

Moreover, the behaviour of a complete time-invariant linear dynamical system can always be described by a set of finite-dimensional linear equations.

Theorem 2.1.7 ([120]) *Let us consider the system* $\Sigma = (\mathbb{T}, \mathbb{R}^q, \mathcal{B})$ *with* $\mathbb{T} = \mathbb{N}_0$. *Then there exists a polynomial matrix* R *with real coefficients such that*

$$\mathcal{B} = \{w : \mathbb{T} \to \mathbb{R}^q | R(\sigma)w = 0\}$$

iff Σ *is linear, time-invariant, and complete, i.e. iff its behaviour* $\mathcal{B}$ *is linear, shift invariant (i.e.,* $\sigma\mathcal{B} = \mathcal{B}$*), and closed in the topology of pointwise convergence in* $\mathbb{R}^q$.

Moreover, this theorem implies that any dynamical system described by time-invariant linear difference equations is complete. Nonlinear dynamical systems, in turn, do not have any uniform description, but we still can use some ideas developed for linear systems. In particular, following the same chain of thought as in the proof of Lemma 2.1.6 (see [120, Prop. 4]) one can prove the following general result:

Lemma 2.1.8 *Consider* $\Sigma = (\mathbb{T}, \mathbb{R}^q, \mathcal{B})$. *The system* Σ *is complete if* $\mathcal{B}$ *is closed in the topology of pointwise convergence.*

Proof. *Let* $w : \mathbb{T} \to \mathbb{R}^q$ *be such that* $w|_{[t_1,t_2]} \in \mathcal{B}|_{[t_1,t_2]}$ *for all* $t_1, t_2 \in \mathbb{T}$, $t_1 \leq t_2$. *We have to show that* $w \in \mathcal{B}$. *Let us consider two sequences:* $\{\tau_i\}$ *such that* $\tau_i \in \mathbb{T}, i \in \mathbb{N}$ *which monotonically tends to the minimal element of* $\mathbb{T}$ *as* $i \to \infty$, *and* $\{\theta_k\}$ *such that* $\theta_k \in \mathbb{T}, k \in \mathbb{N}$ *and* $\lim\limits_{k\to\infty} \theta_k = \infty$. *Let* $t_1 = \theta_1 \in \mathbb{T}$. *By assumption there exists* $w_n \in \mathcal{B}$ *such that* $w|_{[\tau_n,\theta_n]} = w_n|_{[\tau_n,\theta_n]}$. *The sequences* w_n *converge pointwise to* w *as* $n \to \infty$. *Since* $\mathcal{B}$ *is closed this implies* $w \in \mathcal{B}$, *as desired.*

Note that this result applies to nonlinear systems evolving not only in discrete, but also in continuous time. Furthermore, the result of Lemma 2.1.8 can be easily extended to the case of a final signal alphabet: $W = \{w_1, \dots\}$. This set can be equipped with the metric

$$\rho(x, y) = \begin{cases} 0, & x = y \\ 1, & x \neq y. \end{cases}$$

This metric induces a topology on the set W which is called discrete topology [85]. Using these constructions one can easily define the notion of pointwise convergence for signals defined on a finite set.

It should be noted, however, that Lemma 2.1.8 gives only a sufficient condition for a nonlinear system to be complete. The following example illustrates this.

Example 2.1.9 *Let us consider* $\Sigma = (\mathbb{T}, \mathbb{R}^q, \mathcal{B})$ *such that for some fixed* $\tau \in \mathbb{T}$

$$\mathcal{B} = \{w : \mathbb{T} \to \mathbb{R}^q | w(t) \in S, w(\tau) = 0, t \neq \tau\},$$

where S in an open set in $\mathbb{R}^q$. The behaviour $\mathcal{B}$ is not closed in the topology of pointwise convergence though it is obviously 0-complete or memoryless.

Incompleteness is a rather rare phenomenon which appears only in particular cases. For instance, a system may be incomplete if its behavioural equations involve operations over the whole time axis as illustrated in the following example.

Example 2.1.10 *Let us consider two behaviours: $\mathcal{B}_1 = \{w \in l_2(\mathbb{N}_0, \mathbb{R}^n) : ||w||_2 < 1)\}$ and $\mathcal{B}_2 = \{w \in l_2(\mathbb{N}_0, \mathbb{R}^n) : ||w||_2 \leq 1)\}$. Both are defined on the Hilbert space of infinite sequences l_2 equipped with the norm $|| \cdot ||_2 = \left(\sum_{i=0}^{\infty} |w(i)|^2 \right)^{\frac{1}{2}}$, where $| \cdot |$ is the standard Euclidean norm. The behaviour $\mathcal{B}_2$ is complete whereas $\mathcal{B}_1$ is not. The latter follows from the fact that it is always possible to find a sequence $\bar{w} \notin \mathcal{B}_1, ||\bar{w}||_2 = 1$ which coincides with sequence $w_k \in \mathcal{B}_1,\ ||w_k||_2 < 1,\ k = 1, \ldots$ on the finite interval $[0, k]$.*

Another example will be considered in Sec. 2.3 in connection with the full and external behaviour of a dynamical system.

Finally, we give a definition of a *trim* dynamical system.

Definition 2.1.11 (Trim system, [121]) *The dynamical system $\Sigma = (\mathbb{T}, W, \mathcal{B})$ is said to be trim if for all $\omega \in W$ there exists $w \in \mathcal{B}$ and $t \in \mathbb{T}$ such that $w(t) = \omega$.*

In a trim system any symbol ω may occur. In most cases the system can be rendered trim by a simple redefinition of the set W.

In the following, we will often need to perform operations on behaviours. The standard set operations, such as union and intersection are defined for behaviours in a usual way, since behaviours are sets. In addition, we introduce two new operations:

- **Concatenation.** Let $w_1, w_2 \in \mathcal{B}$ and $t \in \mathbb{T}$. We define concatenation of w_1 and w_2 at t as

$$(w_1 \wedge_t w_2)(\tau) = \begin{cases} w_1(\tau) & \tau < t \\ w_2(\tau) & \tau \geq t. \end{cases}$$

  The generalisation to concatenation of behaviours is straightforward:

$$\mathcal{B}_1 \wedge_t \mathcal{B}_2 = \{w \in W^{\mathbb{T}} |\ w|_{[0,\tau)} \in \mathcal{B}_1|_{[0,\tau)}\,, w|_{[\tau,\infty)} \in \mathcal{B}_2|_{[\tau,\infty)}\}.$$

- **Projection.** Let $\mathcal{B} \subseteq (W_1 \times W_2)^{\mathbb{T}}$. Projection $\mathcal{P}_{W_1}\mathcal{B}$ of $\mathcal{B}$ on the set W_1 is defined as follows:

$$\mathcal{P}_{W_1}\mathcal{B} = \{w_1 \in W_1^{\mathbb{T}} | \exists w_2 \in W_2^{\mathbb{T}}, (w_1, w_2) \in \mathcal{B}\}.$$

  Let $\mathcal{B}' \subseteq W_1^{\mathbb{T}}$. The right inverse[1] of the projection operator is defined as

$$\mathcal{P}_{W_1}^{-1}\mathcal{B}' = \{(w_1, w_2) \in (W_1 \times W_2)^{\mathbb{T}} | w_1 \in \mathcal{B}'\}.$$

Often it is advantageous to consider signals as elements of a certain functional space, especially if the signal space $W = \mathbb{R}^n$. From now on, if not stated otherwise, we assume that all discrete time signals $w \in W^{\mathbb{N}_0}$ are defined in the space $\ell^\infty(\mathbb{N}_0, W)$ and all continuous signals $w \in W^{\mathbb{R}_{\geq 0}}$ belong to the set of locally integrable functions $\mathcal{L}_1^{\text{loc}}(\mathbb{R}_{\geq 0}, W)$. More details on the development of behavioural concepts applied to specific classes of dynamical systems can be found in [95].

2.2 Systems with inputs and outputs

Up to now, we assumed that there aren't any distinctions among the single components of the signal. However, in many dynamical systems there are variables that can be freely assigned whereas the remaining ones cannot. Hence, we need to define a specific class of variables, namely *locally free variables*.

Definition 2.2.1 (Locally free variables, [121]) *Let $\Sigma = (\mathbb{T}, W_1 \times W_2, \mathcal{B})$ be a dynamical system. We say that the variable w_1 is locally free if $\mathcal{P}_{W_1}\mathcal{B}$ is trim and memoryless.*

The following example illustrates the difference between these two types of variables.

Example 2.2.2 *Let us consider the discrete time dynamical system $\Sigma = (\mathbb{N}_0, W_1 \times W_2, \mathcal{B})$, $W_1 = W_2 = \mathbb{R}$ whose behaviour is defined in the following way:*

$$\mathcal{B} = \{(w_1, w_2) \in (\mathbb{R} \times \mathbb{R})^{\mathbb{N}_0} | w_2(k+1) = w_2(k) \cdot \sin(w_1(k)),\ k \in \mathbb{N}_0\}.$$

Obviously, the signal w_1 is locally free, i.e., $\mathcal{P}_{W_1}\mathcal{B} = W_1^{\mathbb{N}_0}$, whereas the signal w_2 belongs to the set

$$\mathcal{W}_2 = \{w_2 \in W_2^{\mathbb{N}_0} : |w_2(i+1)| \leq |w_2(i)|, i \in \mathbb{N}_0\} = \mathcal{P}_{W_2}\mathcal{B} \subset W_2^{\mathbb{N}_0}.$$

[1] Note that $\mathcal{P}_W \mathcal{P}_W^{-1} \equiv Id$, whereas, in general, $\mathcal{P}_W^{-1}\mathcal{P}_W \not\equiv Id$.

The locally free variables can be considered now as natural candidates for inputs. But it is still not sufficient for a system to have an input/output structure. It has to be examined whether the outputs do not anticipate the inputs.

Definition 2.2.3 (Non-anticipation, [121]) *Consider* $\Sigma=(\mathbb{T}, W_1\times W_2, \mathcal{B})$. *We say that* w_2 *does not anticipate* w_1 *if*

$$\begin{aligned}&\left(\mathcal{P}_{W_1}\tilde{w}|_{[0,\tau]} = \mathcal{P}_{W_1}\hat{w}|_{[0,\tau]}\right) \Rightarrow \\ &\qquad \Rightarrow (\exists w \in \mathcal{B})\left[\mathcal{P}_{W_2}w|_{[0,\tau]} = \mathcal{P}_{W_2}\tilde{w}|_{[0,\tau]} \text{ and } \mathcal{P}_{W_1}w = \mathcal{P}_{W_1}\hat{w}\right]\end{aligned}$$

for all $\tilde{w}, \hat{w} \in \mathcal{B}, \tau \in \mathbb{T}$. *Moreover, we say that* w_2 *does strictly not anticipate* w_1 *if the premise in the above implication can be weakened to* $\{\mathcal{P}_{W_1}\tilde{w}|_{[0,\tau)} = \mathcal{P}_{W_1}\hat{w}|_{[0,\tau)}\}$.

In the following, we will consider dynamical systems satisfying both requirements. They have free variables (which we call inputs) that are not anticipated by the remaining variables (which, in turn, we call outputs). The definition of such systems is given below.

Definition 2.2.4 (I/- system, [78]) *The system* $\Sigma = (\mathbb{T}, W_1 \times W_2, \mathcal{B})$ *is said to be I/- w.r.t.* (W_1, W_2) *if:*

1. w_1 *is locally free;*
2. w_2 *does not anticipate* w_1.

Moreover, the system Σ *is said to be strictly I/- w.r.t.* (W_1, W_2) *if the following conditions hold:*

1. w_1 *is locally free;*
2. w_2 *does strictly not anticipate* w_1.

We will call W_1 *and* W_2 *input and output sets and the signals* w_1 *and* w_2 *input and output, respectively.*

Obviously, a strict I/- system is an I/- system.

From now on we will use the symbols U and u to denote the input set and the input signal and the symbols Y and y to denote the output set and the output signal. Sometimes, however, this may lead to some confusion, especially in the case of interconnected systems where the output of one system can be considered as the input of the other. In these cases, if it is not clear from the context, we will explicitly say that the system is (strictly) I/- with respect to $(\cdot, \cdot)$, where the first component refers to the input and the second one to the output.

One might notice that the definition of an I/- system is a slightly weakened version of the definition of Willems' I/O system. The difference is that the former one does not require the output to process the input. This requirement implies that there exists an operator F which maps the input signal u to the output signal y (see [121, Th. 3.1]).

2.3 State systems

2.3.1 Dynamical systems in state space form

In the previous sections we implicitly assumed that the behaviour of a dynamical system consists of only those signals which can be immediately observed, i.e. of external signals. However, it is often convenient to include internal variables in behavioural description as well. The following example illustrates this.

Example 2.3.1 *Let us consider an electro-magnetic circuit consisting of a coil with an iron core shown in Fig. 2.3a). The behavioural equation is*

$$\frac{d}{dt}\Phi + Ri = v,$$

where the magnetic flux Φ *depends on the magnetisation of the iron core, i.e., on the position of the working point on the hysteresis curve (see Fig. 2.3b). Thus, in order to describe the behaviour of this electromagnetic system one has to take into consideration the value of the internal variable* Φ*.*

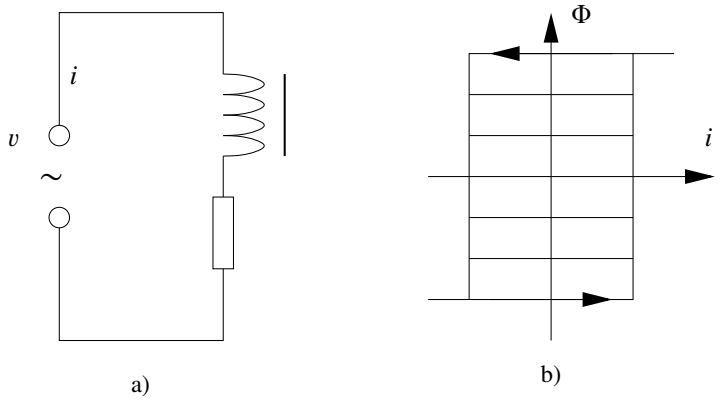

Figure 2.3: Electro-magnetic circuit

We have seen that sometimes we have to consider not only those variables through which the system "communicates" with its environment, but also some internal variables (like the magnetic flux in the case of the electromagnetic circuit). These variables are called *latent variables*.

Definition 2.3.2 (Latent variables, [121]) *A dynamical system with latent variables is a quadruple*

$$\Sigma_a = (\mathbb{T}, W, A, \mathcal{B}_a),$$

where $\mathbb{T}$ is the time axis, W is the signal alphabet, A is the space of latent variables and $\mathcal{B}_a \subseteq (W \times A)^{\mathbb{T}}$ is the (extended) behaviour.

Moreover, there is one serious drawback which is typical for the input/output behavioural description of dynamical systems: one has to analyse finite (but sometimes rather lengthy) fragments of the input/output signals, which often appears to be very inefficient. Hence, one might ask whether it is possible to find some quantities which would instantaneously characterise the system. It is, indeed, possible to use latent variables for this purpose if they satisfy the *axiom of state*.

Definition 2.3.3 (Dynamical system in state space form, [121]) *Let $\Sigma_s = (\mathbb{T}, W, X, \mathcal{B}_s)$ be a dynamical system with latent variables $x \in X^{\mathbb{T}}$. We will call Σ_s a dynamical system in state space form and X state space if the following implication holds:*

$$\{(w_1, x_1), (w_2, x_2) \in \mathcal{B}_s, x_1(\tau) = x_2(\tau), \tau \in \mathbb{T}\} \Rightarrow \{(w_1, x_1) \wedge_\tau (w_2, x_2) \in \mathcal{B}_s\}.$$

This implication is called the axiom of state. The state of the system at time τ together with the future external signals completely characterises the future evolution of the system and can be seen as a representative of the entire interval $(w, x)|_{[0,\tau)\cap\mathbb{T}}$.

Any state system can be efficiently represented by a *state machine*. There are different definitions of a state machine for discrete and continuous time, but the difference is mainly in notation (cf. discrete and continuous time evolution laws in [121, Sec. 1.5]):

Definition 2.3.4 (Discrete time state machine) *A discrete time state machine is a tuple $P_d = (X, W, \delta, X_0)$, where W, X and $X_0 \subseteq X$ denote the external signal space, the state space, and the set of initial conditions, $\delta \subseteq X \times W \times X$ is the transition (next state) relation.*

Definition 2.3.5 (Continuous time state machine) *A continuous time state machine is a tuple $P_d = (X, W, \delta, X_0)$, where W is the external signal space, X and $X_0 \subseteq X$ are the state space (differential manifold) and the set of initial conditions, $\delta \subseteq TX \times W$ is the transition (vector field) relation. $TX = \coprod_{x\in X} T_xX$ denotes the tangent bundle of X, where $\coprod_{x\in X} T_xX$ is the disjoint union of tangent spaces T_xX to X at points $x \in X$.*

In the following, we will consider only the discrete time case to keep our notation simple. However, most results that will be presented in the sequel can be extended for the continuous time case at the cost of some additional notations. One serious argument for this choice is the fact that the continuous time case admits many particular cases whose consideration can be rather tedious, whereas we want to keep the exposition as straightforward as possible. Finally, notice that the discrete time is natural for all real life (read "digitally controlled") applications and hence, does not lead to a serious narrowing of the scope.

State machines can be classified with respect to their state space.

Definition 2.3.6 (Finite and hybrid state machines) *A state machine $P = (W, X, \delta, X_0)$ is called finite if $|X| < \infty$. The state space of a hybrid state machine is a product $X = D \times \mathbb{R}^n$, where $1 < |D| < \infty$.*

Finally, we have to establish an interrelation between behavioural and state machine representations of a dynamical system. Given a state machine $P = (W, X, \delta, X_0)$, the behaviour

$$\mathcal{B}_s = \{(w, x) \in (W \times X)^{\mathbb{N}_0} | (x(k), w(k), x(k+1)) \in \delta \ \forall k \in \mathbb{N}_0, \ x(0) \in X_0)\}$$

is referred to as the *induced full behaviour*, and $\Sigma_s = (\mathbb{N}_0, W \times X, \mathcal{B}_s)$ as the induced state space system. The projection $\mathcal{B} = \mathcal{P}_W \mathcal{B}_s$ is called the *external behaviour* of the system Σ_s. A state machine P' with external behaviour $\mathcal{B}'$ is said to be a *realisation* of the dynamical system $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$. This is denoted by $P' \simeq \Sigma'$ [81].

The external behaviour can be found as a result of the *state elimination* procedure, which has been formalised for different classes of dynamical systems, e.g., linear time-invariant differential systems [75, 123] and differential-algebraic systems [31, 116]. For nonlinear systems, there is the notion of external differential representation which seeks for a representation of the nonlinear system as a set of high-order differential equations in the inputs and outputs [86, Sec. 4.2].

In the converse case, the external behaviour $\mathcal{B}$ is given and one has to find a state space representation $\Sigma_s = (\mathbb{N}_0, W \times X, \mathcal{B}_s)$ (or its realisation $P \simeq \Sigma_s$) such that $\mathcal{P}_W \mathcal{B}_s = \mathcal{B}$. This problem is called the *realisation* problem. The realisation problem has been extensively studied for over 50 years, first for finite automata (Myhill-Nerode theorem [56]) then for linear [100] and nonlinear differential systems (generating series, in particular Fliess' series [117, 119]). Later, these results were generalised for hybrid systems [45]. For the classification of state maps within the behavioural systems theory see the recent paper by Julius and van der Schaft [53].

It is worth noting that the external behaviour does not always possess the properties of the full behaviour. One important example is the completeness property.

Example 2.3.7 *Let us consider the discrete time state machine P whose transition relation δ is described by the following equations*

$$\begin{aligned} x(k+1) &= \alpha x(k) \\ y(k) &= \begin{cases} 1, & x(k) \geq 1 \\ 0, & x(k) < 1 \end{cases} \end{aligned} \tag{2.1}$$

$$X_0 = X = \mathbb{R}_{\geq 0},\ k \in \mathbb{N}_0,\ \alpha \in (0,1).$$

The induced full behaviour of P is $\mathcal{B}_s = \{(x,y) \in (X \times Y)^{\mathbb{N}_0} | (2.1) \text{ holds}\}$. The external behaviour $\mathcal{B} = \mathcal{P}_Y \mathcal{B}_s$ is - as follows from (2.1) - a set of sequences of the form $(1, \dots, 1, 0, \dots)$, where the number of 1-elements is finite (possibly zero). However, it can easily be seen that the sequence $(1, \dots, 1, \dots)$ consisting only of 1-elements cannot be distinguished from those belonging to $\mathcal{B}$ through the analysis of strings of finite length. For any $k \in \mathbb{N}_0$ there exists $x \in X_0$ such that the resulting sequence of output symbols contains exactly k 1-elements. Hence, the external behaviour $\mathcal{B}$ is not complete!

At the same time one can check that there does not exist any signal $(x,y) \in (X \times Y)^{\mathbb{N}_0}$ which matches the induced full behaviour $\mathcal{B}_s$ on all fixed length intervals but does not belong to $\mathcal{B}_s$.

2.3.2 I/S/- machines

By analogy with the previous section, we can define a state dynamical system equipped with an input/output structure. As in the previous section, we consider a slightly weakened version of Willems' I/S/O system, namely an I/S/- system:

Definition 2.3.8 (I/S/- system, [78]) *An I/S/- system is a tuple*

$$\Sigma_{I/S/-} = (\mathbb{N}_0, U, Y, X, \mathcal{B}_{I/S/-}),$$

where U is the input alphabet, Y is the output alphabet, X is the state set, and $\mathcal{B}_{I/S/-} \subseteq (U \times Y \times X)^{\mathbb{N}_0}$ is the system's behaviour which satisfies the following conditions:

i. $\mathcal{B}_{I/S/-}$ satisfies the axiom of state, i.e.,

$$\begin{aligned} &\{(u_1,y_1,x_1),(u_2,y_2,x_2) \in \mathcal{B}_{I/S/-},\ x_1(\tau) = x_2(\tau),\ \tau \in \mathbb{N}_0\} \Rightarrow \\ &\quad \Rightarrow \{(u_1,y_1,x_1) \wedge_\tau (u_2,y_2,x_2) \in \mathcal{B}_{I/S/-}\}; \end{aligned}$$

ii. u is locally free;

iii. in $\mathcal{B}_{I/S/-}$, u is strictly not anticipated by x and not anticipated by y.

As in the case of I/- systems, we do not require the state x and the output y to process the input signal u.

An I/S/- dynamical system can be represented by a (slightly modified) state machine. The main difference is that, as in the case of I/S/- behavioural systems, we distinguish between inputs and outputs.

Definition 2.3.9 (I/S/- machine) *The tuple*

$$P_{I/S/-} = (U, Y, X, \delta, X_0)$$

where

- *U, Y, X are the input, output, and state sets;*
- *$X_0 \subseteq X$ is the set of initial states;*
- *$\delta : X \times U \times Y \times X$ is the transition relation,*

is said to be an I/S/- machine if for each reachable state $\xi \in X$ and for each $\eta \in U$ there exist $\xi' \in X$ and $\upsilon \in Y$ such that $(\xi, \eta, \upsilon, \xi') \in \delta$.

The state $\tilde{\xi} \in X$ is said to be reachable if there exists an initial state $\xi_0 \in X_0$, a constant $\kappa \in \mathbb{N}_0$, and sequences $\{\eta_i\}_{i=0,\kappa-1}$, $\{\upsilon_i\}_{i=0,\kappa-1}$, and $\{\xi_i\}_{i=1,\kappa}$ such that

$$(\xi_j, \eta_j, \upsilon_j, \xi_{j+1}) \in \delta \quad \forall j = 0, \kappa - 1$$

and $\xi_\kappa = \tilde{\xi}$.

It should be noted that a state machine is, in principle, an untimed model. Hence, we need to determine some rules which would allow us to assign the input/output symbols to the respective points in time. We associate the predecessor and the successor states with the time instants $t = k$ and $t' = k + 1$, where $k \in \mathbb{N}_0$. When assigning the input/output symbols one has to take into account the requirements of Def. 2.3.8. Therefore, we associate both η and υ symbols in the transition relation δ with the time $t = k$. Now we can define the induced full behaviour of the state machine $P_{I/S/-}$ as

$$\begin{aligned}\mathcal{B}_P = \{(u, y, x) \in (U \times Y \times X)^{\mathbb{N}_0} \mid \\ ((x(k), u(k), y(k), x(k+1)) \in \delta \, \forall k \in \mathbb{N}_0, \; x(0) \in X_0\}.\end{aligned}$$

It can be easily proved that the behaviour $\mathcal{B}_P$ of the state machine $P_{I/S/-}$ does indeed satisfy the axioms of Def. 2.3.8. The future evolution of a state

machine depends only on the current state and the current + future values of the input signal; the input is locally free and, finally, the input u is strictly not anticipated by x and not anticipated by y.

The transition relation δ is not well suited for analysis. One often needs a more convenient description. Therefore, we define a transition function $\gamma : X \times U \to 2^{Y \times X}$ as

$$\gamma(\xi, \eta) = \{(\upsilon, \xi') \in Y \times X | (\xi, \eta, \upsilon, \xi') \in \delta\}.$$

Moreover, we assume that the successor state ξ' and the output signal υ are independent:

A3. The transition function $\gamma(\xi, \eta)$ can be represented as

$$\gamma(\xi, \eta) = \gamma_y(\xi, \eta) \times \gamma_x(\xi, \eta), \tag{2.2}$$

where $\gamma_y : X \times U \to 2^Y$ and $\gamma_x : X \times U \to 2^X$.

This assumption can be interpreted in two ways. On the one hand, we require that the current output of the system does not depend on the future evolution of the state, i.e., that y does not anticipate x. This is absolutely natural. On the other hand, we assume that the state evolution does not depend on the output y. This, again, turns out to be very natural, since the evolution of the state is a composition of its internal and forced dynamics. Neither of them is subjected to the influence of the system output, which merely reflects them.

Finally, we are ready to write down the behavioural equations corresponding to an I/S/- state machine:

$$\begin{cases} x(k+1) \in \gamma_x(x(k), u(k)), \\ y(k) \in \gamma_y(x(k), u(k)), \quad k \in \mathbb{N}_0 \\ x(0) \in X_0. \end{cases} \tag{2.3}$$

Note that the functions $\gamma_x(x(k), u(k))$ and $\gamma_y(x(k), u(k))$ are defined for all $\eta \in U$ and for all reachable states $\xi \in X$. Furthermore, in virtue of Def. 2.3.9 for all $\eta \in U$ and for all reachable states $\xi \in X$ the sets $\gamma_x(\xi, \eta)$ and $\gamma_y(\xi, \eta)$ are not empty.

We say that the behavioural equations are time-invariant if their right-hand sides are invariant w.r.t. the transformation $k' = k + \Delta k,\ \Delta k \in \mathbb{N}_0$. This holds as time k does not appear as an argument in function γ. Furthermore, the full induced behaviour of an I/S/- machine is time-invariant if its behavioural equations are time invariant and $X_0 = X$. It can be easily shown that this is also true for the external behaviour.

In an I/S/- machine the output y does not anticipate the input u. Let us now consider one special case. Let the function γ_y depend only on the first argument:

$$\gamma_y(\xi, \eta) = \tilde{\gamma}_y(\xi) \ \forall \eta \in U, \tag{2.4}$$

where $\tilde{\gamma}_y(\xi) : X \to 2^Y$. In this case the output y *strictly* does not anticipate the input u. In the sequel we will call an I/S/- machine *strictly non anticipating* if (2.4) holds.

The following observation is straightforward:

Lemma 2.3.10 *The external behaviour $\mathcal{B} = \mathcal{P}_{(U\times Y)}\mathcal{B}_P$ of a (strictly) non-anticipating I/S/- machine P is a (strict) I/- behaviour.*

In conclusion, we consider two classes of finite automata with inputs and outputs.

Example 2.3.11 (Moore and Mealy automata) *A finite I/S/- state machine with behavioural equations (2.3) is called a (non-deterministic) Mealy automaton. If, additionally, condition (2.4) is satisfied, the state machine is said to be a (non-deterministic) Moore automaton. The corresponding automata are called deterministic if the functions γ_x and γ_y are single-valued, and the set X_0 is a singleton.*

In Fig. 2.4 an example of Moore and Mealy automata with $U = \{a, b, c\}$, $Y = \{\alpha, \beta\}$, $X = \{0, 1, 2\}$, and $X_0 = \{0\}$ is shown. The formal definition and further details can be found in [20, Sec. 2.2.5].

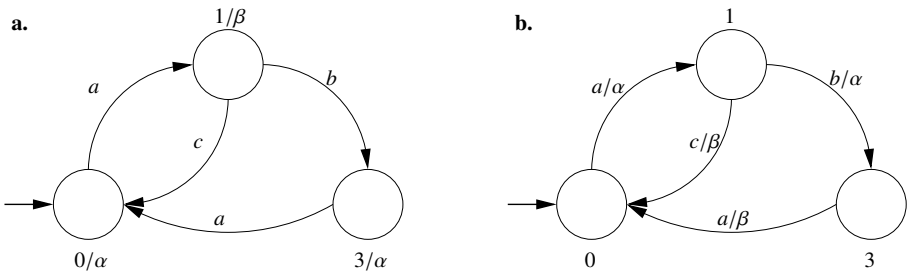

Figure 2.4: Moore (a.) and Mealy (b.) automata

2.4 Interconnection of dynamical systems

This section is devoted to the analysis of the interconnection of two dynamical systems. We define the closed-loop behaviour of interconnected systems and investigate its properties.

There are two ways to consider the interconnection of two dynamical systems: a behavioural description and a constructive representation based

on state models. In the following, we employ both of them since they form a well established framework where the reasoning is based on the behavioural description, and computations are carried out with the help of state models.

2.4.1 Behavioural description

Let us first consider the composition (interconnection) of two dynamical systems with respect to their external behaviours as shown in Fig. 2.5. The resulting system admits only those signals that satisfy the behavioural equations of both systems.

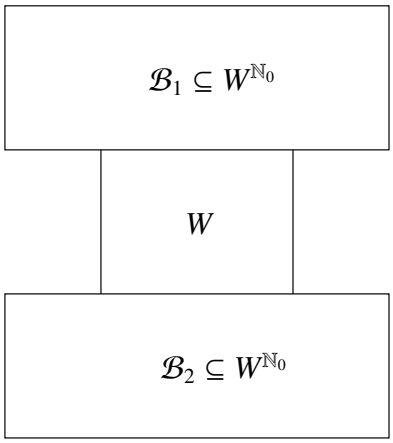

Figure 2.5: Interconnection of two dynamical systems

Definition 2.4.1 *Let* $\Sigma_1=(\mathbb{N}_0, W, \mathcal{B}_1 \subseteq W^{\mathbb{N}_0})$ *and* $\Sigma_2=(\mathbb{N}_0, W, \mathcal{B}_2 \subseteq W^{\mathbb{N}_0})$ *be two dynamical systems. The composition of* Σ_1 *and* Σ_2 *is defined as*

$$\Sigma_1 \times \Sigma_2 := (\mathbb{N}_0, W, \mathcal{B}_1 \cap \mathcal{B}_2).$$

The interconnection of two dynamical systems corresponds to the intersection of their external behaviours. Furthermore, if these systems possess an I/O structure, we need to specify a way in which the input and output of one system are connected to the input and output of the other one. Thus, we define a special case of composition operation: *feedback interconnection.*

Definition 2.4.2 *Let* $\Sigma_1 = (\mathbb{N}_0, U_1, Y_1, \mathcal{B}_1)$ *and* $\Sigma_2 = (\mathbb{N}_0, U_2, Y_2, \mathcal{B}_2)$ *be two I/- dynamical systems. Let* $U_1 = Y_2$ *and* $U_2 = Y_1$*. The composition of* Σ_1 *and* Σ_2 *presented in Fig. 2.6 is called* feedback interconnection *and denoted by* $\Sigma_1 \times \Sigma_2$.

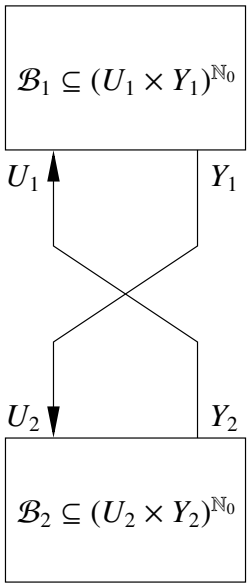

Figure 2.6: Feedback interconnection

Here and throughout, we consider the feedback interconnection of dynamical systems, i.e., the case where the output of one system is connected to the input of the other one and vice versa.

In the following, we introduce a property which will play a crucial role in the analysis of interconnected systems, namely, the notion of *non-conflicting behaviours*.

Definition 2.4.3 ([78]) *Two behaviours $\mathcal{B}_1, \mathcal{B}_2 \subseteq (U \times Y)^{\mathbb{N}_0}$ are said to be* non-conflicting *if*

$$\mathcal{B}_1|_{[0,k]} \bigcap \mathcal{B}_2|_{[0,k]} = (\mathcal{B}_1 \cap \mathcal{B}_2)|_{[0,k]} \text{ for all } k \in \mathbb{N}_0. \tag{2.5}$$

In words, if a signal agrees with both $\mathcal{B}_1$ and $\mathcal{B}_2$ on the finite time interval $[0, k]$, it can be extended to the whole time axis such that the resulting signal belongs to $\mathcal{B}_1 \cap \mathcal{B}_2$. The opposite situation, where the signal "gets stuck" at some time instant is called *blocking*. This happens if the equality relation in the latter expression is replaced by the "proper subset" relation: $\mathcal{B}_1|_{[0,k]} \bigcap \mathcal{B}_2|_{[0,k]} \supset (\mathcal{B}_1 \cap \mathcal{B}_2)|_{[0,k]}$ for some $k \in \mathbb{N}_0$.

Non-conflictingness is the fundamental property which ensures the consistency of the interconnection of two dynamical systems. We illustrate this by a simple example.

Example 2.4.4 *Let us consider the feedback interconnection of two dynamical systems Σ_1 and Σ_2 represented by two Mealy automata P_1 and P_2 with*

$X_{1,2} = \{0, 1\}$, $U_1 = Y_2 = \{a, b\}$, *and* $Y_1 = U_2 = \{\alpha, \beta\}$. *The structure of* P_1 *and* P_2 *is shown in Fig. 2.7. The respective external behaviours are:*

$$\mathcal{B}_1 = \overline{\Big(\overline{(b/\beta)}\ a/\alpha\ \overline{(a/\beta)}\ b/\alpha\Big)}, \quad \mathcal{B}_2 = \overline{\Big(\overline{(\beta/b)}\ \alpha/a\ \overline{(\alpha/a)}\ \beta/b\Big)},$$

where $\overline{(a/\alpha)}$ *is a (possibly infinite) concatenation of symbols* a *and* α. *We assume that all elements of* $\mathcal{B}_i$, $i = 1, 2$ *are infinite sequences. The intersections of two behaviours* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *is*

$$\mathcal{B}_1 \bigcap \mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2 = \begin{pmatrix} \overline{(b)} \\ \overline{(\beta)} \end{pmatrix},$$

where the signals u_2 *and* y_2 *in the second behaviour* $\mathcal{B}_2$ *had been permuted to ensure correctness of the intersection operation, i.e.,*

$$\mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2 = \{(y_2, u_2) \in (Y_2 \times U_2)^{\mathbb{N}_0} | (u_2, y_2) \in \mathcal{B}_2\}.$$

Now let us check the non-conflictingness condition (2.5) for $k = 0$*:*

$$\Big(\mathcal{B}_1 \bigcap \mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2\Big)\Big|_{[0,0]} = \begin{pmatrix} b \\ \beta \end{pmatrix},$$

$$\mathcal{B}_1|_{[0,0]} \bigcap \mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2|_{[0,0]} = \left(\begin{pmatrix} b \\ \beta \end{pmatrix}, \begin{pmatrix} a \\ \alpha \end{pmatrix} \right).$$

Hence, we can observe that

$$\mathcal{B}_1 \bigcap \mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2 \Big|_{[0,0]} \neq \mathcal{B}_1|_{[0,0]} \bigcap \mathcal{P}_{Y_2 \times U_2} \mathcal{B}_2|_{[0,0]}$$

and, therefore, the feedback interconnection of the two dynamical systems Σ_1 *and* Σ_2 *is conflicting.*

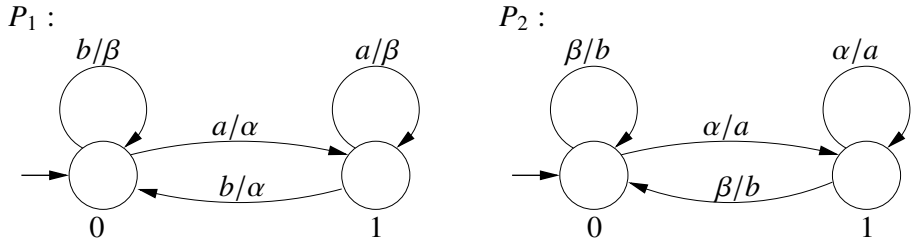

Figure 2.7: Two Mealy automata

There is also one specific situation which sometimes occurs in practice.

Definition 2.4.5 *Two behaviours $\mathcal{B}_1, \mathcal{B}_2 \subseteq (U \times Y)^{\mathbb{N}_0}$ are said to be* incompatible *if $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset$ and* compatible *otherwise.*

Note that two incompatible behaviours are non-conflicting. The following example illustrates this phenomenon.

Example 2.4.6 *Two memoryless dynamical systems, whose dynamics are described by equations:*

$$\begin{aligned} \Sigma_1 : \quad & y(k) = f(u(k)) \\ \Sigma_2 : \quad & u(k) = g(y(k)), \quad k \in \mathbb{T} \end{aligned}$$

are incompatible if the fixed point problem $y = f \circ g(y)$ does not have any solution.

Let us consider two simple systems: $\Sigma_1 : \ y(k) = \frac{1}{2}u(k) + 1$, $\Sigma_2 : \ u(k) = 2y(k)$. Obviously, there does not exist any y which satisfies the equation $y = y + 1$. Hence, Σ_1 and Σ_2 are incompatible.

Otherwise, if there is at least one solution of the fixed point problem, the systems are compatible and non-conflicting. Notice that since both systems are memoryless and time-invariant, one has to check the non-conflictingness property only for $k = 0$.

We want to stress here that non-conflictingness as well as incompatibility is always defined in terms of external behaviours. Even if we consider I/S/-dynamical systems, only external signals matter. In the following, we will say that two dynamical systems Σ_1 and Σ_2 are non-conflicting (incompatible) if their external behaviours $\mathcal{B}_1$, $\mathcal{B}_2$ are non-conflicting (incompatible).

2.4.2 Interconnection of state space dynamical systems

The interconnection of two I/S/- dynamical systems can be characterised in a similar way as in Def. 2.4.1:

Definition 2.4.7 *Let Σ_1=$(\mathbb{N}_0, U_1, Y_1, X_1, \mathcal{B}_1)$ and Σ_2=$(\mathbb{N}_0, U_2, Y_2, X_2, \mathcal{B}_2)$ be two I/S/- dynamical systems with $U_1 = Y_2$ and $Y_1 = U_2$. The composition of Σ_1 and Σ_2 is defined as*

$$\Sigma_1 \times \Sigma_2 := (\mathbb{N}_0, U, Y, X, \mathcal{B}),$$

where $U = U_1 = Y_2$, $Y = U_2 = Y_1$, $X \subseteq X_1 \times X_2$, and $\mathcal{B} \subseteq (U \times Y \times X)^{\mathbb{N}_0}$ such that

$$\mathcal{P}_{U \times Y}\mathcal{B} = \mathcal{P}_{U_1 \times Y_1}\mathcal{B}_1 \cap \mathcal{P}_{Y_2 \times U_2}\mathcal{B}_2.$$

Note that $\Sigma_1 \times \Sigma_2$ is not necessarily I/- w.r.t. (U, Y).

Assume that both dynamical systems are represented by I/S/- machines, i.e., $\Sigma_1 \simeq P_1{=}(U_1, Y_1, X_1, \delta_1, X_{0,1})$, $\Sigma_2 \simeq P_2{=}(U_2, Y_2, X_2, \delta_2, X_{0,2})$. Then, the behaviour of the composite system can be described as

$$\mathcal{B} = \{(u, y, x_1, x_2) \in (U \times Y \times X)^{\mathbb{N}_0} |\ (2.6) \text{ holds } \}$$

$$\begin{cases} (x_1(k), u(k), y(k), x_1(k+1)) \in \delta_1 \\ (x_2(k), y(k), u(k), x_2(k+1)) \in \delta_2 \\ x_1(0) \in X_{0,1},\ x_2(0) \in X_{0,2}\ , \end{cases} \tag{2.6}$$

where U and Y are defined as in Def. 2.4.7.

Now we are ready to describe requirements on dynamical systems to be non-conflicting.

Theorem 2.4.8 *Let Σ_1 and Σ_2 be two I/S/- dynamical systems with external behaviours $\mathcal{B}_1$ and $\mathcal{B}_2$ which are represented by I/S/- machines P_1 and P_2. Moreover, assume that P_1 and P_2 satisfy the requirement (2.2) of assumption* **A3.** *Then, Σ_1 and Σ_2 are non-conflicting if either of the following fixed point inclusions is satisfied for all reachable $(\xi_1, \xi_2) \in (X_1 \times X_2)$:*

$$\begin{aligned} \upsilon_1 &\in \gamma_{y,1}(\xi_1, \gamma_{y,2}(\xi_2, \upsilon_1)) \\ \upsilon_2 &\in \gamma_{y,2}(\xi_2, \gamma_{y,1}(\xi_1, \upsilon_2)). \end{aligned} \tag{2.7}$$

Proof. *To prove non-conflictingness we have to show that for any string $\vartheta_k \in (U \times Y)^{k+1}$, $k \in \mathbb{N}_0$ such that $\vartheta_k \in \mathcal{B}_1|_{[0,k]} \cap \mathcal{P}_{U\times Y}\mathcal{B}_2|_{[0,k]}$ there exists a sequence $(u, y) \in (U \times Y)^{\mathbb{N}_0}$ such that $\vartheta_k = (u, y)|_{[0,k]}$ and $(u, y) \in (\mathcal{B}_1 \cap \mathcal{P}_{U\times Y}\mathcal{B}_2)$.*

Let such a ϑ_k be given. Then, there exist two sequences $\zeta_1 \in (U \times Y \times X_1)^{k+1}$ and $\zeta_2 \in (U \times Y \times X_2)^{k+1}$ such that $\zeta_1 = (u, y, x_1) \in \mathcal{B}_s(P_1)|_{[0,k]}$ and $\zeta_2 = (u, y, x_2) \in \mathcal{P}_{U\times Y\times X_2}\mathcal{B}_s(P_2)|_{[0,k]}$, where $\mathcal{B}_s(P_i)$ is the full induced behaviour of P_i, $i = 1, 2$. Moreover, $\mathcal{P}_{(U\times Y)}\zeta_1 = \mathcal{P}_{(U\times Y)}\zeta_2 = \vartheta_k$.

It should be shown that ζ_1 and ζ_2 can be extended to $\tilde{\zeta}_1 \in \mathcal{B}_s(P_1), \tilde{\zeta}_2 \in \mathcal{P}_{U\times Y\times X_2}\mathcal{B}_s(P_2)$ such that $\mathcal{P}_{(U\times Y)}\tilde{\zeta}_1 = \mathcal{P}_{(U\times Y)}\tilde{\zeta}_2$.

The evolution of the interconnected system of the two state machines P_1 and P_2 is described by the following set of inclusions:

$$\begin{cases} x_1(k+1) &\in \gamma_{x,1}(x_1(k), u(k)), \\ y(k) &\in \gamma_{y,1}(x_1(k), u(k)), \\ x_2(k+1) &\in \gamma_{x,2}(x_2(k), y(k)), \\ u(k) &\in \gamma_{y,2}(x_2(k), y(k)), \quad k \in \mathbb{N}_0 \\ (x_1(0), x_2(0)) &\in X_{0,1} \times X_{0,2}. \end{cases} \tag{2.8}$$

Let $\xi_1 \in X_1$ be a reachable state of P_1 and $\xi_2 \in X_2$ be a reachable state of P_2. The sets $\gamma_{x,1}(\xi_1, \eta)$ and $\gamma_{x,_2}(\xi_2, \upsilon)$ are not empty for all $\eta \in U$ and $\upsilon \in Y$ by virtue of Def. 2.3.9.

The values of input and output signals at time $k+1$ have to satisfy the following set of inclusions:

$$\begin{cases} y(k+1) & \in & \gamma_{y,1}(x_1(k+1), u(k+1)), \\ u(k+1) & \in & \gamma_{y,2}(x_2(k+1), y(k+1)), \end{cases}$$

which are reciprocally related - the value of one function is the argument for the other, and vice versa. Hence, we can rewrite them as fixed point inclusions (2.7), which are in fact equivalent. If any of them is solvable for all reachable states $\xi_1 \in X_1$ and $\xi_2 \in X_2$, there always exists a one-step prolongation. Then, the obtained values for $x_1(k+1)$, $x_2(k+1)$, $u(k+1)$, and $y(k+1)$ can be used to calculate the next step and so forth. □

The conditions of Theorem 2.4.8 can be checked for certain classes of systems, for instance, for systems described by linear equations. To illustrate this we consider a simple example which is devoted to a classical problem.

Example 2.4.9 *Let us consider the case of feedback interconnection of two linear discrete-time systems Σ_1 and Σ_2 whose dynamics is described by the following time-invariant behavioural equations:*

$$\Sigma_1 : \quad \begin{cases} x_1(k+1) = A_1 x_1(k) + B_1 u(k) \\ y(k) = C_1 x_1(k) + D_1 u(k) \end{cases}$$

$$\Sigma_2 : \quad \begin{cases} x_2(k+1) = A_2 x_2(k) + B_2 y(k) \\ u(k) = C_2 x_2(k) + D_2 y(k) \end{cases}$$

where $x_1(k), x_2(k) \in \mathbb{R}^n, u(k) \in \mathbb{R}^m, y(k) \in \mathbb{R}^l, k \in \mathbb{N}_0$, $(x_1(0), x_2(0)) \in X_0 \subseteq (\mathbb{R}^n \times \mathbb{R}^n)$, and A_i, B_i, C_i, D_i, $i = 1, 2$ are matrices of appropriate dimensions. The systems Σ_1 and Σ_2 are non-conflicting if matrix $(I_{(l \times l)} - D_1 D_2)$ is non-singular, i.e., if

$$rank(I - D_1 D_2) = l.$$

Otherwise, the corresponding behaviours are conflicting.

Note that the systems Σ_1 and Σ_2 become non-conflicting if at least one of the two matrices D_1 and D_2 is set equal to zero. The system $\tilde{\Sigma}_i$, modified in this way, is strictly non-anticipating whereas the initial system Σ_i is just non-anticipating.

In practice, in most cases it is impossible to check whether the inclusions (2.7) are solvable for all states. Therefore, we need a possibly more restrictive but practically more useful condition. In the previous example, we have seen that the property of a system to be non-anticipating or strictly non-anticipating may play a crucial role. The following theorem justifies this observation and gives a general, structural condition for two dynamical systems to be non-conflicting.

Theorem 2.4.10 *Let Σ_1 and Σ_2 be two trim I/S/- dynamical systems which can be represented by I/S/- machines P_1 and P_2. Moreover, assume that P_1 and P_2 satisfy the requirement (2.2) of assumption* **A3.** *Then, Σ_1 and Σ_2 are non-conflicting if at least one of the two state machines P_1 and P_2 is strictly non-anticipating (i.e., if condition (2.4) holds).*

Proof. *In this case the fixed point inclusions are transformed to trivial inclusions which are satisfied per definition. E.g., if the first system is strictly non-anticipating, the second inclusion in (2.8) takes the form*

$$y(k) \in \tilde{\gamma}_{y,1}(x_1(k))$$

and the corresponding fixed point inclusion looks as follows:

$$y_2 \in \gamma_{y,2}(x_2, \tilde{\gamma}_{y,1}(x_1)).$$

□

Note that the order in which we consider the behaviours is not important.

2.5 Supervisory control problem

Classical control theory considers the following question: *"how to influence a dynamical system (plant) to achieve the desired behaviour?"* A solution of this problem is usually represented in the form of another dynamical system (the controller) which, when connected to the plant, ensures the fulfilment of the requirements imposed on the closed-loop system. The supervisory control approach, in turn, deals with a more general problem: *"how to restrict the behaviour of a dynamical system to the least extent in order to satisfy a given set of constraints?"* Note that this does not imply the existence of a unique controller. The supervisory control system restricts the plant behaviour by excluding solely those signals which violate the given specification. Hence, in supervisory control we must operate with sets of signals rather than with single ones.

Let us assume that there is a set of safety constraints given by the behaviour $\mathcal{B}_{sp} \subseteq (U \times Y)^{\mathbb{N}_0}$ which contains all allowable signals, i.e., those

signals that do not violate the constraints. The behaviour $\mathcal{B}_{sp}$ is called *specification*. The supervisory control problem can be formulated as follows:

Definition 2.5.1 *Let the plant be given as an I/- system* $\Sigma_{pl} = (\mathbb{N}_0, U, Y, \mathcal{B}_{pl})$ *with external behaviour* $\mathcal{B}_{pl}$ *and the specification be defined as* $\mathcal{B}_{sp} \subseteq (U \times Y)^{\mathbb{N}_0}$. *The I/- system* $\Sigma_{sup} = (\mathbb{N}_0, Y, U, \mathcal{B}_{sup})$ *is said to be a solution of the supervisory control problem* $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ *if for the feedback interconnection* $\Sigma_{pl} \times \Sigma_{sup} = (\mathbb{N}_0, U \times Y, \mathcal{B}_{pl} \cap \mathcal{B}_{sup})$ *the following holds:*

i. $\mathcal{B}_{pl} \cap \mathcal{B}_{sup} \subseteq \mathcal{B}_{sp}$,

ii. Σ_{pl} *and* Σ_{sup} *are non-conflicting.*

A solution of $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ *is called a* supervisor.

In the following, we will often identify the supervisory control problem $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ and the set of all solutions to this problem. Hence, the expression $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp} = \{\emptyset\}$ means that there are no solutions to $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ except the trivial one, i.e., $\mathcal{B}_{sup} = \emptyset$.

The set of all solutions of the supervisory control problem is often infinite and includes the trivial case $\Sigma_{sup,\emptyset} = (\mathbb{N}_0, Y, U, \emptyset)$. The following lemma establishes the existence of the maximal element of this set which is called a *"maximally permissive" supervisor*.

Lemma 2.5.2 ([78]) *Let* A *be an index set and* $\Sigma_\alpha = (\mathbb{N}_0, Y, U, \mathcal{B}_\alpha)$ *be a solution of* $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ *for all* $\alpha \in A$. *Then,* $\Sigma_{sup}^{max} = (\mathbb{N}_0, Y, U, \bigcup_{\alpha \in A} \mathcal{B}_\alpha)$ *is also a solution of* $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$.

Supervisor design is a highly nontrivial problem. Here, we just give an outline of the design procedure. The first step consists in the definition of the specification. In many cases we are given a set of safety constraints like, e.g., "the temperature must belong to some interval", "the output valve must not be open while the tank is being filled", or "the rate of change of the input voltage must not exceed a certain value". These constraints impose restrictions on the system outputs, inputs, or both. Let us assume that $\mathcal{B}_f \subset (U \times Y)^{\mathbb{N}_0}$ consists of all signals which violate at least one constraint. Then, the specification can be found as $\mathcal{B}_{sp} = (U \times Y)^{\mathbb{N}_0} \backslash \mathcal{B}_f$. In many practically relevant cases, when the cardinality of both U and Y is finite one can describe the specification as a finite automaton (for details see [20, Sec. 3.3.1]).

The next step is the design of the maximally permissive supervisor that solves the supervisory control problem $(\mathcal{B}_{pl}, \mathcal{B}_{sp})_{cp}$ [99].

Chapter 3

Hierarchical Control

Now we are ready to introduce the key concept of the thesis, namely the idea of hierarchical control architecture. Everybody who has ever had to control a complex technical system knows that it is practically impossible to solve the whole problem at once. There are two main challenges: the structural complexity of the plant and the intricate character of the specification. But if we study the problem intently we may see that there is often some inhomogeneity. The system dynamics can be decomposed into slow and fast modes, the specification includes long and short term tasks as well as local and global restrictions. Hence, it is advantageous to decompose the overall problem into several sub-problems and solve them separately. In this way there appears some natural hierarchy of the problems - at the higher level we solve global, long term problems using abstract (e.g., discrete-event) models whereas at lower levels we are mainly concerned with short term problems, which have to be solved in order to fulfil the needs of the high-level controller.

Hierarchical decomposition is a hardly formalisable problem which requires a very good qualitative knowledge of the system dynamics along with a good piece of engineering intuition. But despite, or rather because of this complexity it makes sense to develop some general framework for the hierarchical systems design. Our main intention is to provide an engineer with a set of instruments and concepts which can (potentially) be used for the solution of complex technical problems. Note that hierarchical structures considered below are closely related to both multilevel and multilayer hierarchies described in Sec. 1.1.2.

This chapter is structured as follows: in Sec. 3.1 we consider a two-level hierarchical control problem and give the definition of a two-level hierarchical solution to the supervisory control problem. Sec. 3.2 describes a bottom-up approach to the design of hierarchical control schemes. Admissibility conditions are discussed in Sec. 3.3. In Sec. 3.4 and 3.5, we analyse two basic structures of the intermediate level. Finally, in Sec. 3.6, we generalise

obtained results to the case of a multi-level control hierarchy.

This chapter develops the ideas proposed by J. Raisch and T. Moor (see [82, 97] and references therein). The contribution of this chapter consists in developing a constructive framework within which it is possible to describe and analyse different hierarchical control structures in an efficient and uniform way. Moreover, we give a detailed description of several possible control structures and analyse their properties with the methods developed in this chapter.

Henceforth, if not otherwise stated, all dynamical systems are assumed to evolve in discrete time, i.e., $\mathbb{T} = \mathbb{N}_0$.

3.1 Two-level hierarchical control architecture

We start our investigation with the analysis of a two-level hierarchical control problem. Despite its apparent simplicity, this problem is well suited to illustrate all substantial features of the hierarchical control concept. First we introduce basic notations and give the definition of a hierarchical supervisory control problem.

Let us consider the two-level control architecture as shown in Fig. 3.1. The plant Σ_{pl} with external behaviour $\mathcal{B}_{pl}^{L} \subseteq (U_L \times Y_L)^{\mathbb{N}_0}$ is assumed to be an I/S/- system w.r.t. (U_L, Y_L). The intermediate layer Σ_{im} with external behaviour $\mathcal{B}_{im} \subseteq (W_L \times W_H)^{\mathbb{N}_0}$, $W_L = U_L \times Y_L$, $W_H = U_H \times Y_H$ "mediates" between the plant and the high-level controller (supervisor) Σ_{sup} with external behaviour $\mathcal{B}_{sup}^{H} \subseteq (Y_H \times U_H)^{\mathbb{N}_0}$, Σ_{sup}, in turn, is assumed to be I/S/- w.r.t. (Y_H, U_H). The indices L and H are introduced to show that the respective behaviour is defined over the set of low-level (W_L) or high-level (W_H) signals. The specification $\mathcal{B}_{sp}^{L} \subseteq W_L^{\mathbb{N}_0}$ is defined in the same way as in Sec. 2.5.

The intermediate layer can implement many different tasks: it can perform spatial as well as temporal discretisation, switch between different (low-level) local controllers in order to meet the requests of the high-level supervisor or manipulate the plant dynamics in a certain way.

Up to now, we did not make any assumptions about the structure of the intermediate level Σ_{im}. However, it is obvious that it must possess certain internal structure to be compatible with both the plant and the high-level supervisor. Hence, before we start to analyse the intermediate level in more details we define the general framework, i.e., the generic class of systems which can be considered as candidates for Σ_{im}. To denote this class we will borrow a term from electrical engineering, namely we will call these systems *quadripoles*.

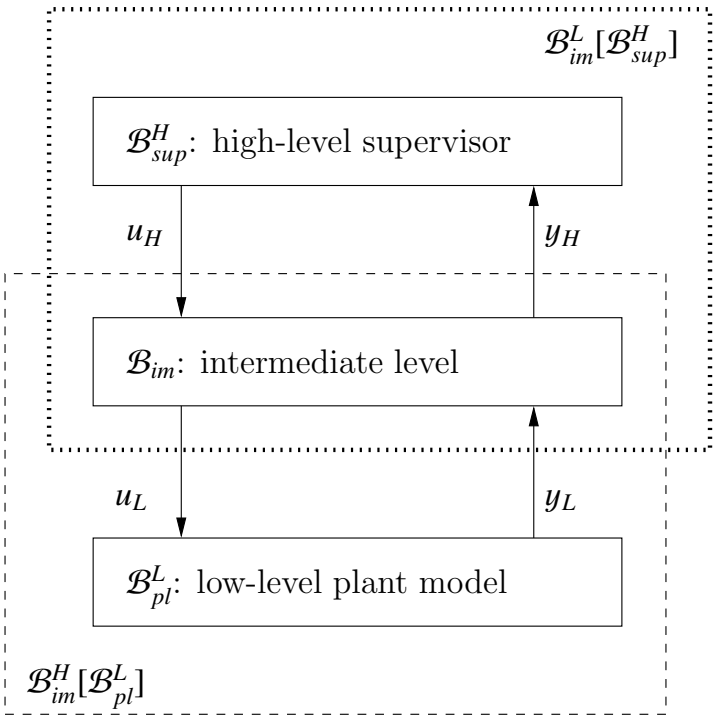

Figure 3.1: Hierarchical control architecture

Definition 3.1.1 (Quadripole) *A quadripole is a tuple*

$$\Sigma_{quad} = (\mathbb{N}_0, U_H, Y_H, U_L, Y_L, X, \mathcal{B}_{quad}),$$

where $W_H = (U_H \times Y_H)$ is the high-level signal space, $W_L = (U_L \times Y_L)$ is the low-level signal space, X is the (hybrid) state space, and $\mathcal{B}_{quad} \subseteq (U_H \times Y_H \times U_L \times Y_L \times X)^{\mathbb{N}_0}$ is the system behaviour which satisfies the following conditions:

i. $\mathcal{B}_{quad}$ satisfies the axiom of state, i.e.,

$$\{(w_1, x_1), (w_2, x_2) \in \mathcal{B}_{quad},\ x_1(\tau) = x_2(\tau),\ \tau \in \mathbb{N}_0\} \Rightarrow \{(w_1, x_1) \wedge_\tau (w_2, x_2) \in \mathcal{B}_{quad}\},$$

where $w = (w_H, w_L) \in (W_H \times W_L)^{\mathbb{N}_0}$;

ii. u_H is locally free;

iii. u_H is strictly not anticipated by x and not anticipated by y_H and u_L;

iv. y_L is locally free;

v. y_L is strictly not anticipated by x and not anticipated by y_H and u_L.

Henceforth, we will assume that the intermediate layer Σ_{im} is a quadripole.

In the following, we will often consider the restriction of the behaviour $\mathcal{B}_{im}$ to a single high-level (low-level) signal. Thus, for $w_H \in W_H^{\mathbb{N}_0}$, we define $\mathcal{B}_{im}^L[w_H]$ as

$$\mathcal{B}_{im}^L[w_H] = \{w_L \in W_L^{\mathbb{N}_0} | (w_H, w_L) \in \mathcal{B}_{im}\}.$$

The restriction $\mathcal{B}_{im}^H[w_L]$ is defined accordingly.

The above definition can be extended to include the restriction of $\mathcal{B}_{im}$ to the sets $\mathcal{B}_{pl}^L$ and $\mathcal{B}_{sup}^H$, denoted by $\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L]$ and $\mathcal{B}_{im}^L[\mathcal{B}_{sup}^H]$, respectively. From the perspective of the plant the intermediate layer and the high-level supervisor form a composite (low-level) supervisor over W_L with external behaviour (see Fig. 3.1)

$$\mathcal{B}_{im}^L[\mathcal{B}_{sup}^H] = \{w_L \in W_L^{\mathbb{N}_0} | (\exists w_H \in \mathcal{B}_{sup}^H)[(w_H, w_L) \in \mathcal{B}_{im}]\}.$$

If, in turn, we look "downwards" we find a similar picture. From the perspective of the high-level supervisor both the intermediate layer and the plant form a composite (high-level) plant over W_H with external behaviour

$$\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L] = \{w_H \in W_H^{\mathbb{N}_0} | (\exists w_L \in \mathcal{B}_{pl}^L)[(w_H, w_L) \in \mathcal{B}_{im}]\}.$$

It may happen that either (or both) of the restrictions $\mathcal{B}_{im}^L[\mathcal{B}_{sup}^H]$ and $\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L]$ do not exist. To prevent this situation, we have to extend Def. 2.4.3, describing the conditions for non-conflicting feedback interconnection of two dynamical systems, to the feedback interconnection of a dynamical system and a quadripole.

Definition 3.1.2 (Non-conflictingness) *Let Σ be a dynamical system with external behaviour $\mathcal{B}^L \subseteq W_L^{\mathbb{N}_0}$ ($\mathcal{B}^H \subseteq W_H^{\mathbb{N}_0}$) and Σ_{im} be a quadripole with external behaviour $\mathcal{B}_{im} \subseteq (W_H \times W_L)^{\mathbb{N}_0}$. The systems Σ and Σ_{im} are said to be non-conflicting if for any $\tilde{u}_H \in U_H^{\mathbb{N}_0}$ there exists $\tilde{y}_H \in P_{Y_H}\mathcal{B}_{im}$ (for any $\tilde{y}_L \in Y_L^{\mathbb{N}_0}$ there exists $\tilde{u}_L \in P_{U_L}\mathcal{B}_{im}$) such that the behaviours $\mathcal{B}^L$ and $\mathcal{B}_{im}^L[(\tilde{u}_H, \tilde{y}_H)]$ $\left(\mathcal{B}^H \text{ and } \mathcal{B}_{im}^H[(\tilde{y}_L, \tilde{u}_L)]\right)$ are non-conflicting.*

Now we can give a formal definition of a two-level hierarchical solution to the supervisory control problem. This definition can be seen as an extension to the standard supervisory control solution from Def. 2.5.1.

Definition 3.1.3 (cf. [82]) *Let $\mathcal{B}_{pl}^L$ be I/- w.r.t. (U_L, Y_L), $\mathcal{B}_{sup}^H$ be I/- w.r.t. (Y_H, U_H) and $\mathcal{B}_{im}$ be the external behaviour of a quadripole with input $(U_H \times Y_L)$ and output $(Y_H \times U_L)$. We say that the pair $(\mathcal{B}_{im}, \mathcal{B}_{sup}^H)_{tl}$ is a two-level hierarchical solution to the supervisory control problem $(\mathcal{B}_{pl}^L, \mathcal{B}_{sp}^L)_{cp}$ if*

i. $\mathcal{B}_{pl}^L \cap \mathcal{B}_{im}^L[\mathcal{B}_{sup}^H] \subseteq \mathcal{B}_{sp}^L$,

ii. $\mathcal{B}_{im}$ *and* $\mathcal{B}^L_{pl}$ *are non-conflicting,*

iii. $\mathcal{B}_{im}$ *and* $\mathcal{B}^H_{sup}$ *are non-conflicting,*

iv. $\mathcal{B}^L_{im}[\mathcal{B}^H_{sup}]$ *is I/- w.r.t.* (Y_L, U_L) *and* $\mathcal{B}^L_{im}[\mathcal{B}^H_{sup}]$ *and* $\mathcal{B}^L_{pl}$ *are non-conflicting, and*

v. $\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}]$ *is I/- w.r.t.* (U_H, Y_H) *and* $\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}]$ *and* $\mathcal{B}^H_{sup}$ *are non-conflicting.*

Here, the first condition guarantees that the hierarchical system does satisfy the low-level specification $\mathcal{B}^L_{sp}$. The remaining four conditions guarantee non-conflicting (non-blocking) functioning of all levels. Note that Definition 3.1.3 contains two more conditions than that in [82]. This is because of the fact that in our work the non-conflicting property is a structural condition formulated in terms of state machines. Hence, we have to ensure that the composite systems $\mathcal{B}^L_{im}[\mathcal{B}^H_{sup}]$ and $\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}]$ can be described as non-blocking state machines which is guaranteed by the requirements *ii.-iii.* Later, in Sec. 3.3, we will see that some of the conditions *ii.–v.* are redundant.

Comparing Def. 2.5.1 and Def. 3.1.3 we can formulate the following statement, which demonstrates an inherent interrelation of hierarchical and monolithic (single-level) supervisory control schemes.

Corollary 3.1.4 *Let* $(\mathcal{B}_{im}, \mathcal{B}^H_{sup})_{tl}$ *be a two-level hierarchical solution to the supervisory control problem* $(\mathcal{B}^L_{pl}, \mathcal{B}^L_{sp})_{cp}$. *Then,* $\mathcal{B}^L_{im}[\mathcal{B}^H_{sup}]$ *is a solution to the supervisory control problem* $(\mathcal{B}^L_{pl}, \mathcal{B}^L_{sp})_{cp}$. *Moreover, if we define*

$$(\mathcal{B}_{im}, \mathcal{B}^H_{sup})^L_{tl} = \left\{ \mathcal{B}^L_{im}[\mathcal{B}^H_{sup}] \subseteq W_L^{\mathbb{N}_0} \,\middle|\, (\mathcal{B}^L_{im}, \mathcal{B}^H_{sup})_{tl} \in (\mathcal{B}^L_{pl}, \mathcal{B}^L_{sp})_{cp} \right\},$$

the following inclusion holds:

$$(\mathcal{B}_{im}, \mathcal{B}^H_{sup})^L_{tl} \subseteq (\mathcal{B}^L_{pl}, \mathcal{B}^L_{sp})_{cp}.$$

Thus, hierarchical solution, as regarded, is a special solution to the supervisory control problem. We will see that the hierarchical approach can drastically simplify the control design procedure. This will be demonstrated in the subsequent sections of this thesis.

3.2 Bottom-up design strategy

A two-level supervisory control problem consists of two sub-problems: the intermediate level design and the high-level supervisor design. In this section we will discuss different approaches to these problems and propose an

abstraction based design strategy which substantially simplifies the overall problem.

Suppose we have designed the intermediate layer Σ_{im} with external behaviour $\mathcal{B}_{im}$. Then, we can reformulate the low-level specification $\mathcal{B}^L_{sp}$ in terms of high-level signals as

$$\mathcal{B}^H_{sp} = \{w_H | (\forall w_L \in \mathcal{B}^L_{pl})[(w_H, w_L) \in \mathcal{B}_{im} \Rightarrow w_L \in \mathcal{B}^L_{sp}]\}.$$

The high-level supervisor could be obtained as a solution to $(\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}], \mathcal{B}^H_{sp})_{cp}$. This approach, however, has one serious drawback. Since the calculation of both $\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}]$ and $\mathcal{B}^H_{sp}$ involves operations with realisations for both plant and specification, it becomes extremely computationally expensive. Moreover, a realisation for $\mathcal{B}^H_{im}[\mathcal{B}^L_{pl}]$, i.e., for the plant under low-level control, is in all likelihood even more complex than a realisation of the plant alone.

Fortunately, high-level supervisor design can be performed on the basis of abstractions, hence it is possible to eliminate from the procedure of high-level supervisor design any operations that directly involve the plant behaviour. For this purpose, we define the behaviour $\mathcal{B}^{HL}_{sp}$, which describes the *intended* relationship between high-level and low-level signals. This specification can be considered from different viewpoints.

On the one hand, the behaviour $\mathcal{B}^{HL}_{sp}$ specifies which information about the low-level process is transmitted to the higher level. This information is the output signal y_H of the composite system plant plus intermediate layer. Note that the system response may be non-deterministic. This means that the system may produce a set of output signals y_H as a reaction to one specific high-level signal u_H. This may be caused by different reasons: non-determinism of the plant, restricted resources of local controllers, imprecise information about plant dynamics and so on.

On the other hand, it describes the "allowable" (or "desirable") reaction of the plant to high-level command signals. This means that, given a high-level input signal u_H, the plant behaviour has to be restricted to be within the set of allowable low-level signals associated to u_H.

This is ensured by the intermediate layer $\mathcal{B}_{im}$ which is designed to implement $\mathcal{B}^{HL}_{sp}$. It satisfies the following requirement:

$$\mathcal{B}_{im} \cap \mathcal{P}^{-1}_{W_L} \mathcal{B}^L_{pl} \subseteq \mathcal{B}^{HL}_{sp}. \tag{3.1}$$

Here we face a typical trade-off. The goal of a designer is to balance the freedom of the low-level and the high-level. Obviously, if we reduce the freedom at the low-level too strongly, it may become impossible to realise a suitable intermediate controller. On the other hand, if it is not restricted sufficiently, we may be unable to find a solution to the high-level supervisory control problem.

Note that, if the design of $\mathcal{B}_{im}$ has been successful, $\mathcal{B}_{sp}^{HL}$ can be seen as an outer approximation of $\mathcal{B}_{im}$ when the latter is connected to the plant $\mathcal{B}_{pl}^{L}$. Thus, we can use $\mathcal{B}_{sp}^{HL}$ to "model" the plant plus intermediate layer. The projection $\tilde{\mathcal{B}}_{pl}^{H} = \mathcal{P}_{W_H}\mathcal{B}_{sp}^{HL}$ can now be used as an abstraction of the high-level plant $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$. Moreover, we define the high-level specification as

$$\tilde{\mathcal{B}}_{sp}^{H} = \{w_H | (\forall w_L)[(w_H, w_L) \in \mathcal{B}_{sp}^{HL} \Rightarrow w_L \in \mathcal{B}_{sp}^{L}]\}. \tag{3.2}$$

Finally, the following lemma states that the high-level supervisor obtained as the solution to the supervisory control problem $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp}$ satisfies requirement (i) in Definition 3.1.3.

Lemma 3.2.1 (cf. [80]) *Let $\mathcal{B}_{im}$ satisfy (3.1). Then, for any high-level nontrivial supervisor $\mathcal{B}_{sup}^{H}$ that solves $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp}$, the inclusion $\mathcal{B}_{pl}^{L} \cap \mathcal{B}_{im}^{L}[\mathcal{B}_{sup}^{H}] \subseteq \mathcal{B}_{sp}^{L}$ is satisfied.*

Proof. *Pick any $w_L \in \mathcal{B}_{pl}^{L} \cap \mathcal{B}_{im}^{L}[\mathcal{B}_{sup}^{H}]$. By definition, there exists a $w_H \in \mathcal{B}_{sup}^{H}$ with $(w_H, w_L) \in \mathcal{B}_{im}$. By Eq.(3.1), we have $(w_H, w_L) \in \mathcal{B}_{sp}^{HL}$ and, therefore, $w_H \in \tilde{\mathcal{B}}_{pl}^{H}$. By virtue of Definition 2.5.1, $w_H \in \tilde{\mathcal{B}}_{sp}^{H}$. Finally, Eq.(3.2) implies $w_L \in \mathcal{B}_{sp}^{L}$.* □

In Algorithm 1, the hierarchical control design procedure is formulated in the form of pseudocode algorithms. Here, the specification $\mathcal{B}_{sp}^{HL}$ plays the central role. We first describe how the composite system plant-intermediate level should behave and then try to design an intermediate level controller which will enforce $\mathcal{B}_{sp}^{HL}$.

Algorithm 1 Bottom-up hierarchical control design

Given: $\mathcal{B}_{pl}^{L}$, $\mathcal{B}_{sp}^{L}$

Required: `Solve` $(\mathcal{B}_{pl}^{L}, \mathcal{B}_{sp}^{L})_{tl}$

`Design the specification` $\mathcal{B}_{sp}^{HL}$
`Compute` $\tilde{\mathcal{B}}_{pl}^{H}$ `and` $\tilde{\mathcal{B}}_{sp}^{H}$
`Solve` $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp}$
if $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp} \neq \{\emptyset\}$
  goto 10
else
  repeat
    `Reformulate the specification` $\mathcal{B}_{sp}^{HL}$
  until $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp} \neq \{\emptyset\}$
`Design the intermediate level controller` $\mathcal{B}_{im}$ `such that`

$$\mathcal{B}_{im} \cap \mathcal{P}_{W_L}^{-1}\mathcal{B}_{pl} \subseteq \mathcal{B}_{sp}^{HL}$$

```
      holds
11:   if there exists a B_im ≠ ∅
12:    Problem solved
13:   else goto 7
14:   end if
15: end if
```

Note that the intermediate controller has to satisfy the non-conflictingness conditions of Def. 3.1.3.

In the following sections, we discuss several specific problems that may appear when designing the specification $\mathcal{B}_{sp}^{HL}$.

3.2.1 Modelling issues

First we want to stress the different nature of the intermediate layer $\mathcal{B}_{im}$ and the specification $\mathcal{B}_{sp}^{HL}$. The specification $\mathcal{B}_{sp}^{HL}$ describes the "expected" behaviour of the system intermediate layer plus plant. In some sense it defines a frame within which this composite structure may evolve. Moreover, since $\mathcal{B}_{sp}^{HL}$ is used for the high-level supervisor design, it must be well structured, i.e., it must allow to pose and solve the high-level control problem $(\tilde{\mathcal{B}}_{pl}^{H}, \tilde{\mathcal{B}}_{sp}^{H})_{cp}$ with some standard procedures.

In Chapter 4 we show how the specification $\mathcal{B}_{sp}^{HL}$ can be represented by an I/S/- machine. In [82], it was proposed to model $\mathcal{B}_{sp}^{HL}$ by a linear hybrid automaton. Obviously, proposed models cannot describe all possible hybrid evolutions. So, one might need to use more complex models like, e.g., hybrid I/O automata [71, 72] or behavioural automata [54] which, however, have to be equipped with inputs and outputs.

At the same time, there is no general representation for the intermediate layer. This reflects, in fact, the core idea of the proposed approach. When designing the high-level supervisor, we are dealing with a standard although rather coarse model while at the low-level we use a control system that is designed and adjusted for the specific plant in such a way that the closed-loop behaviour conforms with the desired model. Later, we will show some particular control structures which can be used at the intermediate layer. Note, however, that there are infinitely many possible control structures.

3.2.2 Qualitative behaviour shaping

In this section we consider one special approach to the design of specification $\mathcal{B}_{sp}^{HL}$ which has been shown to be particularly helpful in many applications. Let us assume that the signal set of the plant U_L can be represented as a

Cartesian product $U_L = U_L^1 \times U_L^2$. Now, we define the specification $\mathcal{B}_{sp}^{HL}$ as

$$\mathcal{B}_{sp}^{HL} = \left\{ (u_H, y_H, u_L, y_L) \in (W_H \times W_L)^{\mathbb{N}_0} \;\middle|\; \begin{array}{l} (u_H, y_H) \in \hat{\mathcal{B}}_{sp}^{H} \\ u_L \in \mathcal{U}_L,\ \mathcal{P}_{U_L^1} u_L = u_H, \\ y_L \in Y_L^{\mathbb{N}_0}, \\ y_H(t) = \phi(y_L(t)), t \in \mathbb{N}_0 \end{array} \right\}, \tag{3.3}$$

where

- $W_H = U_H \times Y_H$ and $W_L = U_L \times Y_L$ are the high-level and low-level signal sets, $U_H = U_L^1$,
- $\hat{\mathcal{B}}_{sp}^{H}$ is an I/- behaviour w.r.t. (U_H, Y_H),
- $\mathcal{U}_L \subseteq U_L^{\mathbb{N}_0}$ is the set of admissible low-level signals such that $\mathcal{P}_{U_L^1}\mathcal{U}_L = (U_L^1)^{\mathbb{N}_0}$,
- $\phi : Y_L \to Y_H$ is an output function, which may be chosen to be the identity.

An intermediate layer $\mathcal{B}_{im}$, designed to satisfy the requirement (3.1), functions in the following way: for any high-level signal u_H it implements low-level signals u_L which, being applied to the plant $\mathcal{B}_{pl}^{L}$, result in that the respective high-level output signals conform to $\hat{\mathcal{B}}_{sp}^{H}$. Moreover, the fixing of a particular high-level signal u_H partially determines the choice of low-level signals: $u_L \in \mathcal{U}_L(u_H)$, where $\mathcal{U}_L(u_H) = \{u_L \in \mathcal{U}_L | \mathcal{P}_{U_L^1} u_L = u_H\}$. In other words, we design low-level controller(s) which, given a high-level signal $u_H = u_1$, uses available low-level signals u_2 to ensure that the high-level response of the composite system $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$ belongs to $\hat{\mathcal{B}}_{sp}^{H}$.

Hence, the abstraction of the high-level plant $\tilde{\mathcal{B}}_{pl}^{H}$ is equal to $\hat{\mathcal{B}}_{sp}^{H}$ and the high-level specification is defined as

$$\tilde{\mathcal{B}}_{sp}^{H} = \{(u_H, y_H) \in W_H^{\mathbb{N}_0} | \mathcal{B}_{sp}^{HL}[(u_H, y_H)] \subseteq \mathcal{B}_{sp}^{L}\}.$$

In certain cases, the structure of the low-level specification can be exploited to simplify the design procedure of the low-level controller. In particular, the set of allowable low-level signals $\mathcal{U}_L$ can be chosen to satisfy the specification $\mathcal{B}_{sp}^{L}$ or to facilitate the analysis of the closed-loop system $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$.

In general, we may notice that the approach described above closely resembles the feedback-based abstraction technique presented in Sec. 1.1.3. Actually, our approach can be seen as a feedback-based abstraction in behavioural form. To complete our presentation we show how the behaviour

$\hat{\mathcal{B}}_{sp}^{H}$ can be used to specify certain qualitative properties of the closed-loop system $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$. There are different properties that can be described in this way, e.g., stability, boundedness, controllability, and so forth. Let us consider the case of real-valued high-level signals: $U_H \subseteq \mathbb{R}^m$, $Y_H \subseteq \mathbb{R}^k$. For this case, we give definitions of ℓ_p *input-output stable behaviours* and *asymptotically stable behaviours.*

Definition 3.2.2 (ℓ_p input-output stability, cf. [55, Def. 5.1]) *The behaviour $\mathcal{B} \subset (U \times Y)^{\mathbb{N}_0}$ is ℓ_p stable if there exist a monotonically increasing function $\alpha : [0, \infty) \to [0, \infty)$ and a non-negative constant β such that for any pair $(u, y) \in \mathcal{B}$*

$$||y||_{\ell_p} \leq \alpha(||u||_{\ell_p}) + \beta$$

holds.

It is finite-gain stable if there exist non-negative constants γ and β such that for any pair $(u, y) \in \mathcal{B}$

$$||y||_{\ell_p} \leq \gamma||u||_{\ell_p} + \beta$$

holds.

The parameter p denotes a specific norm and can be chosen within $\{1, \ldots, \infty\}$. However, the most common cases are ℓ_2 and ℓ_∞.

Definition 3.2.3 (Asymptotic stability) *The behaviour $\mathcal{B} \subset (U \times Y)^{\mathbb{N}_0}$ is stable if for any pair $(u, y) \in \mathcal{B}$ there exists a function ϕ of class[1] $\mathcal{K}$ such that*

$$(\exists T \in \mathbb{N}_0 \ s.t. \ u(t) = 0, \, t \geq T) \Rightarrow (y(t) \leq \phi(\|y(T)\|), \, t \geq T)$$

holds. The behaviour $\mathcal{B} \subset (U \times Y)^{\mathbb{N}_0}$ is asymptotically stable if, additionally,

$$\exists T \in \mathbb{N}_0 \ s.t. \ u(t) = 0, \, t \geq T \Rightarrow \lim_{t \to \infty} y(t) = 0.$$

holds.

Later, in Chapters 4 and 5, two applications will be presented. We will show that the qualitative behaviour shaping technique can be efficiently applied to both continuous and hybrid systems. Moreover, in Ch. 5 we will demonstrate a novel approach based on the monotonisation of the system behaviour.

[1] A real-valued function $\phi(x)$ belongs to class $\mathcal{K}$ if it is defined, continuous, and strictly increasing on $x \in [0, \infty]$, and if it vanishes at $x = 0$, i.e., $\phi(0) = 0$ (see [47]).

3.3 Non-conflictingness conditions

In the previous section we presented the bottom-up design procedure which allows us to design the hierarchical control system in a uniform way. We have shown (see Lemma 3.2.1) that the designed hierarchical control scheme respects the low-level specification and, hence, fulfils the first condition of Def. 3.1.3. In this section, we consider different classes of low-level systems and formulate conditions on the intermediate layer and the high-level supervisor to meet the remaining conditions. Moreover, we will show that some of conditions *ii.–v.* are superfluous. Hereby, we will set up the framework for the design of hierarchical control systems.

The system Σ_{im} can be realised by an I/S/- state machine. We say that the tuple

$$P_{im} = ((U_H \times Y_L), (Y_H \times U_L), X_{im}, \delta_{im}, X_0)$$

where

- U_H and Y_H are high-level signal spaces, U_L and Y_L are low-level signal spaces, and X_{im} the state space;
- $X_{im,0} \subseteq X_{im}$ is the set of initial states;
- $\delta_{im} : X_{im} \times U_H \times Y_H \times U_L \times Y_L \times X_{im}$ is the transition relation,

is a realisation of Σ_{im} if for each reachable state $\xi_{im} \in X_{im}$ and for each pair $(\eta_H, \upsilon_L) \in U_H \times Y_L$ there exist $\xi' \in X_{im}$ and a pair $(\upsilon_H, \eta_L) \in Y_H \times U_L$ such that $(\xi, \eta_H, \upsilon_H, \eta_L, \upsilon_L, \xi') \in \delta_{im}$.

In analogy with Def. 2.3.9, one can define the transition function $\gamma^{im} : X_{im} \times U_H \times Y_L \to 2^{Y_H \times U_L \times X_{im}}$ as

$$\gamma^{im}(\xi, \eta_H, \upsilon_L) = \{(\upsilon_H, \eta_L, \xi') \in Y_H \times U_L \times X_{im} | (\xi, \eta_H, \upsilon_H, \eta_L, \upsilon_L, \xi') \in \delta_{im}\}.$$

Furthermore, we assume that the function $\gamma^{im}(\xi, \eta_H, \upsilon_L)$ satisfies Assumption **A3**:

$$\gamma^{im}(\xi, \eta_H, \upsilon_L) = \gamma_y^{im}(\xi, \eta_H, \upsilon_L) \times \gamma_u^{im}(\xi, \eta_H, \upsilon_L) \times \gamma_x^{im}(\xi, \eta_H, \upsilon_L),$$

where $\gamma_y^{im} : X_{im} \times U_H \times Y_L \to 2^{Y_H}$, $\gamma_u^{im} : X_{im} \times U_H \times Y_L \to 2^{U_L}$ and $\gamma_x^{im} : X_{im} \times U_H \times Y_L \to 2^{X_{im}}$. Now we can write down the behavioural equations defining the evolution of P_{im}:

$$\begin{cases} x_{im}(k+1) \in \gamma_x^{im}(x_{im}(k), u_H(k), y_L(k)), \\ y_H(k) \in \gamma_y^{im}(x_{im}(k), u_H(k), y_L(k)), \\ u_L(k) \in \gamma_u^{im}(x_{im}(k), u_H(k), y_L(k)), \quad k \in \mathbb{N}_0 \\ x_{im}(0) \in X_{im,0}. \end{cases} \tag{3.4}$$

Again, it can be seen that the behaviour defined by (3.4) does indeed satisfy the conditions of Def. 3.1.1.

In the system Σ_{im} realised by P_{im}, y_H does strictly not anticipate u_H (y_L) if condition (3.5) (condition (3.6)) holds for all reachable $\xi \in X_{im}$ and for all $\eta_H \in U_H$, $\upsilon_L \in Y_L$:

$$\gamma_y^{im}(\xi, \eta_H, \upsilon_L) = \hat{\gamma}_y^{im}(\xi, \upsilon_L), \tag{3.5}$$

$$\gamma_y^{im}(\xi, \eta_H, \upsilon_L) = \tilde{\gamma}_y^{im}(\xi, \eta_H). \tag{3.6}$$

Similarly, u_L does strictly not anticipate u_H (y_L) if condition (3.7) (condition (3.8)) holds for all reachable $\xi \in X_{im}$ and for all $\eta_H \in U_H$, $\upsilon_L \in Y_L$:

$$\gamma_u^{im}(\xi, \eta_H, \upsilon_L) = \hat{\gamma}_u^{im}(\xi, \upsilon_L), \tag{3.7}$$

$$\gamma_u^{im}(\xi, \eta_H, \upsilon_L) = \tilde{\gamma}_u^{im}(\xi, \eta_H). \tag{3.8}$$

Finally, y_H (u_L) does strictly not anticipate both u_H and y_L, if the corresponding output function depends only on $\xi \in X_{im}$.

Now, we can formulate two lemmata that will play a central role in the subsequent analysis of the non-conflictingness property of hierarchical structures. They are formulated for the interconnection of Σ_{pl} and Σ_{im}, but due to the symmetric structure of the intermediate level Σ_{im}, they can easily be reformulated, mutatis mutandis, for the interconnection of Σ_{sup} and Σ_{im}.

Lemma 3.3.1 *Let Σ_{pl} be an I/S/- dynamical system with external behaviour $\mathcal{B}_{pl}^L \subseteq W_L^{\mathbb{T}}$ and let P_{pl} be an I/S/- machine realising $\mathcal{B}_{pl}^L$. Furthermore, let Σ_{im} be a quadripole with external behaviour $\mathcal{B}_{im}$ realised by a trim state machine P_{im} satisfying assumption* **A3.** *The feedback interconnection of Σ_{pl} and Σ_{im} is non-conflicting if for the machine P_{im} the following condition holds:*

i. u_L does strictly not anticipate y_L, i.e., (3.8) holds.

Then, the behaviour $\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L]$ can be realised by an I/S/- machine. If, additionally, the following conditions hold for P_{im}, the behaviour $\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L]$ can be realised by a strictly non-anticipating state machine:

ii. y_H does strictly not anticipate u_H, i.e., (3.5) holds,

iiia. y_H does strictly not anticipate y_L, i.e., (3.6) holds,
or
iiib. u_L does strictly not anticipate u_H, i.e., (3.7) holds.

Proof. *The proof is similar to those of Theorems 2.4.8 and 2.4.10. We can write down the behavioural equations of the composite system and show that they are well-defined for all values of the high-level input u_H. Let us first consider the composite system $\Sigma_{im}[\Sigma_{pl}]$ when the first condition holds:*

$$\begin{cases} x_{pl}(k+1) \in \gamma_x^{pl}\left(x_{pl}(k), \tilde{\gamma}_u^{im}(x_{im}(k), u_H(k))\right), \\ x_{im}(k+1) \in \gamma_x^{im}\left(x_{im}(k), u_H(k), \gamma_y^{pl}(x_{pl}(k), \tilde{\gamma}_u^{im}(x_{im}(k), u_H(k)))\right), \\ y_H(k) \in \gamma_y^{im}\left(x_{im}(k), u_H(k), \gamma_y^{pl}(x_{pl}(k), \tilde{\gamma}_u^{im}(x_{im}(k), u_H(k)))\right), \ k \in \mathbb{N}_0 \\ x_{pl}(0) \in X_{pl,0}, \ x_{im}(0) \in X_{im,0}. \end{cases} \tag{3.9}$$

The expressions in the right-hand sides of (3.9) are defined for all $u_H(k) \in U_H$ and for all reachable states $x_{pl}(k) \in X_{pl}$ and $x_{im}(k) \in X_{im}$. Moreover, the state (x_{pl}, x_{im}) does strictly non anticipate u_H and y_H does not anticipate u_H. Therefore, the behavioural equations (3.9) describe an I/S/- machine. Herewith we have proved the first part.

Furthermore, from the analysis of the equation for y_H we can directly check the non-anticipation property of the composite system. Let us consider two cases:

Case *i+ii+iiia:*
Condition iiia, together with ii, boils down to $\gamma_y^{im}(x, u_H, y_L) = \bar{\gamma}_y^{im}(x)$.

$$\begin{cases} x_{pl}(k+1) \in \gamma_x^{pl}\left(x_{pl}(k), \tilde{\gamma}_u^{im}(x_{im}(k), u_H(k))\right), \\ x_{im}(k+1) \in \gamma_x^{im}\left(x_{im}(k), u_H(k), \gamma_y^{pl}(x_{pl}(k), \tilde{\gamma}_u^{im}(x_{im}(k), u_H(k)))\right), \\ y_H(k) \in \bar{\gamma}_y^{im}(x_{im}(k)), \quad k \in \mathbb{N}_0 \\ x_{pl}(0) \in X_{pl,0}, \ x_{im}(0) \in X_{im,0}. \end{cases}$$

Case *i+ii+iiib:*
Condition iiib, together with i, boils down to $\gamma_u^{im}(x, u_H, y_L) = \bar{\gamma}_u^{im}(x)$.

$$\begin{cases} x_{pl}(k+1) \in \gamma_x^{pl}\left(x_{pl}(k), \bar{\gamma}_u^{im}(x_{im}(k))\right), \\ x_{im}(k+1) \in \gamma_x^{im}\left(x_{im}(k), u_H(k), \gamma_y^{pl}(x_{pl}(k), \gamma_u^{im}(x_{im}(k)))\right), \\ y_H(k) \in \hat{\gamma}_y^{im}\left(x_{im}(k), \gamma_y^{pl}(x_{pl}(k), \bar{\gamma}_u^{im}(x_{im}(k)))\right), \quad k \in \mathbb{N}_0 \\ x_{pl}(0) \in X_{pl,0}, \ x_{im}(0) \in X_{im,0}. \end{cases}$$

Hence, both cases correspond to strictly non-anticipating I/S/- machines. □

Lemma 3.3.2 *Let Σ_{pl} be an I/S/- dynamical system with external behaviour $\mathcal{B}_{pl}^L \subseteq W_L^{\mathbb{T}}$ which can be represented by a strictly non-anticipating state machine P_{pl}. Furthermore, let Σ_{im} be a quadripole with external behaviour*

$\mathcal{B}_{im}$ realised by a trim state machine P_{im} satisfying assumption **A3.** *The feedback interconnection of Σ_{pl} and Σ_{im} is non-conflicting.*

The behaviour $\mathcal{B}_{im}^H[\mathcal{B}_{pl}]$ can be realised by a state machine. If, additionally, for P_{im} y_H does strictly not anticipate u_H, i.e., (3.5) holds, the behaviour $\mathcal{B}_{im}^H[\mathcal{B}_{pl}]$ can be realised by a strictly non-anticipating state machine.

Proof. *The fact that the behaviour $\mathcal{B}_{im}^H[\mathcal{B}_{pl}]$ can be realised by an I/S/- machine follows from Lemma 3.3.1 as a strictly non-anticipating state machine P_{pl} is a special case of an I/S/- machine. The second part can be proved in analogy with the proof of Lemma 3.3.1. The behavioural equations of the composite system are*

$$\begin{cases} x_{pl}(k+1) \in \gamma_x^{pl}\left(x_{pl}(k), \gamma_u^{im}(x_{im}(k), u_H(k), \bar{\gamma}_y^{pl}(x_{pl}(k))\right), \\ x_{im}(k+1) \in \gamma_x^{im}\left(x_{im}(k), u_H(k), \bar{\gamma}_y^{pl}(x_{pl}(k))\right), \\ y_H(k) \in \hat{\gamma}_y^{im}\left(x_{im}(k), \bar{\gamma}_y^{pl}(x_{pl}(k))\right), \quad k \in \mathbb{N}_0 \\ x_{pl}(0) \in X_{pl,0},\ x_{im}(0) \in X_{im,0}. \end{cases}$$

□

The following theorem states that some of the non-conflictingness conditions are superfluous.

Theorem 3.3.3 *To check the non-conflictingness conditions for a two-level hierarchical structure (conditions ii.–v. of Def. 3.1.3) it suffices to check two conditions:*

C1:
1. *$\mathcal{B}_{im}$ and $\mathcal{B}_{pl}^L$ are non-conflicting, and*
2. *$\mathcal{B}_{im}^H[\mathcal{B}_{pl}^L]$ and $\mathcal{B}_{sup}^H$ are non-conflicting.*

This is equivalent to checking the following two conditions:

C2:
1. *$\mathcal{B}_{im}$ and $\mathcal{B}_{sup}^H$ are non-conflicting, and*
2. *$\mathcal{B}_{im}^L[\mathcal{B}_{sup}^H]$ and $\mathcal{B}_{pl}^L$ are non-conflicting.*

Proof. *A pair of conditions is satisfied if the systems Σ_{pl}, Σ_{im} and Σ_{sup} possess certain structural properties which follow from Theorem 2.4.10 and Lemmata 3.3.1 and 3.3.2. Below, we list all possible combinations for both cases:*

C1:

$\mathcal{B}_{pl}$	$\mathcal{B}_{im}$	$\mathcal{B}_{sup}$
n.a.	*(3.8)*	*s.n.a.*
n.a.	*(3.8), (3.5), (3.6)*	*n.a.*
n.a.	*(3.8), (3.5), (3.7)*	*n.a.*
s.n.a.	*(3.5)*	*n.a*

C2:

$\mathcal{B}_{sup}$	$\mathcal{B}_{im}$	$\mathcal{B}_{pl}$
n.a.	*(3.5)*	*s.n.a.*
n.a.	*(3.5), (3.8), (3.6)*	*n.a.*
n.a.	*(3.5), (3.8), (3.7)*	*n.a.*
s.n.a.	*(3.8)*	*n.a*

where 'n.a.' and 's.n.a' denote non-anticipating and strictly non-anticipating, respectively. These two sets of structural conditions are equivalent. Therefore, the fulfilment of conditions **C1** *implies the fulfilment of conditions* **C2** *and vice versa.* □

Given the plant $\mathcal{B}_{pl}^{L}$, the intermediate layer $\mathcal{B}_{im}$, the high-level supervisor $\mathcal{B}_{sup}^{H}$, and the state machines representing these behaviours, the non-conflictingness conditions of Th. 3.3.3 can be checked using results of Theorem 2.4.10 as well as the results of Lemmata 3.3.1 and 3.3.2.

Since the structure of the plant is given it is natural to check the non-conflictingness conditions bottom-up, i.e., to check conditions **C1**. Note that this is consistent with the bottom-up design strategy proposed above. Thus, given the plant $\mathcal{B}_{pl}^{L}$ and the respective state machine, both the intermediate layer $\mathcal{B}_{im}$ and the high-level supervisor $\mathcal{B}_{sup}^{H}$ must be chosen in such a way that $\mathcal{B}_{im}$ and $\mathcal{B}_{pl}$ are non-conflicting, the composite system $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$ is I/-w.r.t. (U_H, Y_H) and that $\mathcal{B}_{im}^{H}[\mathcal{B}_{pl}^{L}]$ and the high-level supervisor $\mathcal{B}_{sup}^{H}$ are non-conflicting.

Below, we consider two rather general schemes which can be used for the design of the intermediate layer and analyse their properties with regard to the non-conflictingness requirement.

3.4 Interface layer

As was said in Sec. 1.1.2, a typical feature of the hierarchical control architecture is that the plant and the high-level supervisor (or the higher level controller) "live in different worlds". Therefore, we need a layer which would establish a communication between lower and higher layers. Following [59], we will call this layer the *interface layer*, or simply the *interface*[2]. The functions of an interface are threefold:

[2]Note that in [97] it was referred to as *type II intermediate layer*.

1. synchronisation of the high-level and the low-level signals;

2. aggregation of the low-level information and translation of this information in a form suitable for the high-level system;

3. implementation of the high-level signals as the low-level control actions.

Below, we consider these three functions in details and discuss the well-posedness issue. In Sec. 3.4.2, the behavioural description of interface layer is given. Finally, in Sec. 3.4.3 we describe the state machine representations of an interface layer and discuss issues regarding the non-conflictingness conditions.

3.4.1 Interface layer: functioning

The structure of a typical interface layer is shown in Fig. 3.2. In many cases, high-level and low-level signals are defined on different time scales. Typically, low-level signals are defined in continuous time or on a discrete time axis provided by fast equidistant sampling. High-level signals, in turn, are defined on a discrete time axis which may depend on the low-level signal. There are two types of synchronisation between the high-level and the low-level system: *time-driven* and *event-driven* (see [89, 57, 101, 102] for details).

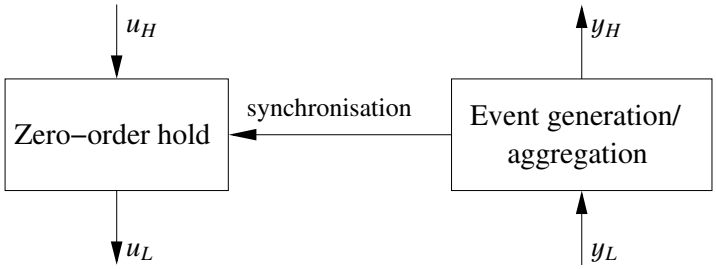

Figure 3.2: An event generator

In the first case, the high-level time scale is equidistant with the sampling time defined as a multiple of the low-level sampling time: $\tau_H = \kappa\tau_L$, where τ_H and τ_L are the high-level and low-level sampling times, and $\kappa \in \mathbb{N}$ is the multiplier. This is the case of *time-driven* high-level dynamics and the corresponding interface layer is said to work as a *sampling unit*.

In the second case, the high-level time axis is generated by low-level signals. This is referred to as *event-driven* high-level dynamics. The interface is said to work as an *event generator*. Event-driven systems produce output signals in response to the occurrence of asynchronous discrete events. These

events are triggered if certain conditions are satisfied. This can be stated formally:

Definition 3.4.1 *We say that the event $\sigma \in \Sigma$ occurs at time θ_i, $i \geq 1$ if*

$$\theta_i = \min_{\sigma \in \Sigma} \{\theta_{i,\sigma}\},$$

where

$$\theta_{i,\sigma} = \min\{t \mid \mathcal{C}_\sigma(\mathbf{y_L}(\theta_{i-1}, t), \theta_{i-1}, t) = true\},$$

$\theta_0 = 0$, $\mathcal{C}_\sigma : \mathcal{Y}_L \times \mathbb{N}_0 \times \mathbb{N}_0 \to \{true, false\}$ is a logical predicate, and $\mathcal{Y}_L = \{y_L(\cdot) \in \ell^\infty([t_1, t_2], Y_L),\ 0 \leq t_1 \leq t_2 < \infty\}$ is a set of the segments of the output signals on intervals $[t_1, t_2] \subset \mathbb{N}_0$. Each particular segment is denoted by $\mathbf{y_L}(t_1, t_2)$.

This means that starting at time $\theta_0 = 0$ the system evolves until some predicate $\mathcal{C}_\sigma$ takes on the value "true" at time τ. At this time instant the event σ occurs. Then, we define $\theta_1 = \tau$ and repeat the procedure. The sequence $\mathcal{T} = \{\theta_0, \theta_1, \dots\}$ defines the high-level time axis.

In most cases, the definition of the set of events is based on the aggregation of the signal space of the low-level system. Let Y_L be the low-level output set and $\mathcal{Y}$ be a set with cardinality less than that of Y_L. A set-valued map $\phi : Y_L \to 2^{\mathcal{Y}}$ such that for every $y \in \mathcal{Y}$ there exists at least one $y_L \in Y_L$ satisfying $y \in \phi(y_L)$ is said to be an A/D map [29]. The map ϕ defines a cover of Y_L, i.e., a set of subsets $A_i \subset Y_L$ such that $\forall i \in \mathcal{Y}, Y_L \supset A_i = \phi^{-1}(i)$. The sets A_i are called the cells of the cover.

If ϕ is a single-valued surjective function, it induces an equivalence relation $\sim_\phi$ and defines the corresponding quotient set $Y_L/\sim_\phi$ (for details, see Sec. 1.1.3). The set of equivalence classes $[j] \subset Y_L/\sim_\phi,\ j \in \mathcal{Y}$ form a partition of the set Y_L (see Fig. 3.3 for the illustration of the difference between a cover and a partition).

In the case of the output space partition, the event condition can be defined as, e.g.,

$$\mathcal{C}_{jk}(i) = [\phi(y_L(t)) \neq \phi(y_L(\theta_{i-1}))] \wedge [\phi(y_L(\theta_{i-1})) = j] \wedge [\phi(y_L(t)) = k]\,,\ j, k \in \mathcal{Y}.$$

The situation when the set of events is based on a cover of Y_L is more involved since there are different ways to define event conditions, that is to decide when a particular cell becomes active. This additional degree of freedom can be exploited to achieve better robustness characteristics of the system as was discussed in [50]. Further discussion on A/D maps can be found in [29], where the relation between dynamic specifications and A/D maps is studied and the notion of refining an A/D map is introduced.

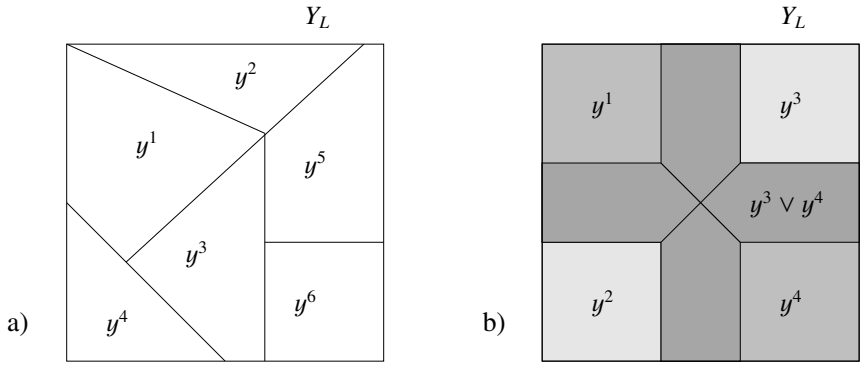

Figure 3.3: A partition a) and a cover b) of the set Y_L

There is also a number of approaches exploiting additional information on the system structure. In particular, a special approach is described in [111, 59], where the partition of the state space is designed based on the natural invariants of the plant.

The high-level signal y_H is produced by an aggregation device which sends a symbol $y_H(k) \in Y_H$ each time an event occurs. The set of high-level output symbols may be equal to the set of events: $Y_H = \Sigma$. An A/D map can also be used to aggregate the low-level information, i.e., $Y_H = \mathcal{Y}$. In this case, the interface layer sends to the higher level the symbol corresponding to the active cell.

High-level control signals u_H, in turn, have to be mapped to low-level control signals u_L. In most cases this is done by implementing a zero-order hold device triggered at the time instants θ_i.

3.4.2 Behavioural description

In this section, we give a formal definition of an interface layer in terms of behaviours.

As we have said above, an interface layer generates the high-level time axis. Obviously, this time axis has to be well defined. Therefore, we introduce a definition of a *time scale*. To do this, we need to extend the notion of causal maps:

Definition 3.4.2 (see, e.g., [55], Ch. 5.1) *An operator $H : U^{\mathbb{N}_0} \to Y^{\mathbb{N}_0}$, i.e. an operator mapping signals u to signals y, is called causal if*

$$\left[\tilde{u}|_{[0,k]} = \hat{u}|_{[0,k]}\right] \Rightarrow \left[H(\tilde{u})|_{[0,k]} = H(\hat{u})|_{[0,k]}\right]$$

holds for all $k \in \mathbb{N}_0$, $\tilde{u}, \hat{u} \in U^{\mathbb{N}_0}$. The operator is said to be strictly causal if

$$\left[\tilde{u}|_{[0,k)} = \hat{u}|_{[0,k)}\right] \Rightarrow \left[H(\tilde{u})|_{[0,k]} = H(\hat{u})|_{[0,k]}\right]$$

for all $k \in \mathbb{N}_0$, $\tilde{u}, \hat{u} \in U^{\mathbb{N}_0}$.

Definition 3.4.3 ([97]) *Let* $T : Y_L^{\mathbb{N}_0} \to \mathbb{N}_0^{\mathbb{N}_0}$. *The operator* T *is said to be a dynamic time scale if* T *is causal and if the time transformation* $T(y_L) : \mathbb{N}_0 \to \mathbb{N}_0$ *is surjective and monotonically increasing for all* $y_L \in Y_L^{\mathbb{N}_0}$.

For a fixed low-level signal y_L, the time transformation $T(y_L)$ maps low-level time $j \in \mathbb{N}_0$ to high-level time $k \in \mathbb{N}_0$. By requiring that T is a causal operator, we ensure that at any instant of time the transformation $T(y_L)$ only depends on the past and current values of y_L. Later, we will also use the pseudo-inverse of the time transformation $T(y_L)$ defined as follows:

$$(T(y_L))^* : \mathbb{N}_0 \to \mathbb{N}_0 \text{ s.t. } (T(y_L))^*(k) = \min_{T(y_L)(j)=k} j, \quad k \in \mathbb{N}_0.$$

This is a monotonically increasing but not surjective mapping. One can easily see that $Im((T(y_L))^*)$ is equal to the sequence of event times $\{\theta_0, \theta_1, \dots\}$ corresponding to the low-level signal y_L.

Often, an additional condition on the dynamic time scale can be imposed. In particular, we may require that the interval between two events cannot exceed some fixed value.

Definition 3.4.4 *Let* $\delta \in \mathbb{N}_0$ *be a constant. A dynamic time scale is called regular modulo* δ *if* $\exists \delta > 0$ *such that*

$$(T(y_L))^*(i+1) - (T(y_L))^*(i) < \delta$$

holds for all $i \in \mathbb{N}_0$ *and for all* $y_L \in Y_L^{\mathbb{N}_0}$.

The regularity requirement can be easily ensured if we define an event $\hat{\sigma}$ with the logical predicate $\mathcal{C}_{\hat{\sigma}}(i) = \{t - \theta_{i-1} \geq \delta\}$. Obviously, the event $\hat{\sigma}$ will occur if no event has occurred during the time δ.

Now let us consider two particular structures of an intermediate layer described in Sec. 3.4.1. In a sampling unit, the time scale is fixed and independent of the low-level output signal y_L, i.e., $T_{su}(y_L) = \tilde{T}_{su}$. $\tilde{T}_{su}$ can be defined as follows:

$$T_{su} : \mathbb{N}_0 \to \mathbb{N}_0 \text{ s.t. } T_{su}(k) = \left\lfloor \frac{k}{\tau} \right\rfloor,$$

where $\tau \in \mathbb{N}$ is a sampling interval, $k \in \mathbb{N}_0$. This is a surjective and monotonic mapping and hence $\tilde{T}_{su}$ is a dynamic time scale.

In the case of an event generator, the operator T depends on y_L. Let $y_L \in Y_L^{\mathbb{N}_0}$ be a low-level signal and $\{\theta_i\}_{i \in \mathbb{N}_0}$ be a corresponding sequence of

event times as described in Def. 3.4.1, the time scale $T_{eg}(y_L) : \mathbb{N}_0 \to \mathbb{N}_0$ is defined by

$$T_{eg}(y_L)(k) = i,\ k \in [\theta_i, \theta_{i+1}),$$

which is obviously surjective and monotonic.

From Def. 3.4.1 we see that the causality requirement holds, i.e., for any $\tilde{y}_L,\ \hat{y}_L \in Y_L^{\mathbb{N}_0}$,

$$\left[\tilde{y}_L|_{[0,k]} = \hat{y}_L|_{[0,k]}\right] \Rightarrow \left[T_{eg}(\tilde{y}_L)|_{[0,k]} = T_{eg}(\hat{y}_L)|_{[0,k]}\right].$$

Thus, an event generator defines a dynamic time scale.

Furthermore, we require the measurement aggregation operator $F : Y_L^{\mathbb{N}_0} \to Y_H^{\mathbb{N}_0}$ to be causal with respect to a dynamic time scale:

Definition 3.4.5 *The operator $F : Y_L^{\mathbb{N}_0} \to Y_H^{\mathbb{N}_0}$ is said to be causal w.r.t. T if T is a dynamic time scale and if*

$$\tilde{y}_L|_{[0,j]} = \hat{y}_L|_{[0,j]} \Rightarrow F(\tilde{y}_L)|_{[0,k]} = F(\hat{y}_L)|_{[0,k]}$$

for $k = T(\tilde{y}_L)(j)$ and all $j \in \mathbb{N}_0$, $\tilde{y}_L, \hat{y}_L \in Y_L^{\mathbb{N}_0}$.

In both cases mentioned above this requirement is fulfilled by construction.

We still have to link high-level control signals u_H to low-level control signals u_L. This is done via a zero-order hold device that is triggered by the time transformation $T(y_L)$, i.e. successive high-level control actions are passed on to the lower level whenever an event is generated and fixed until the next event occurs. Formally, this is expressed by $u_L = u_H \circ T(y_L)$.

In summary, an interface layer $\mathcal{B}_{int}$ mediating between low-level and high-level time is completely defined by a dynamic time scale T and a measurement aggregation operator F that is causal w.r.t. T:

$$\mathcal{B}_{int} = \{(u_H, y_H, u_L, y_L) | y_H = F(y_L) \text{ and } u_L = u_H \circ T(y_L)\}. \tag{3.10}$$

3.4.3 Non-conflictingness conditions

An interface layer is represented by an I/S/- machine

$$P_{int} = ((U_H \times Y_L), (Y_H \times U_L), X_{int}, \gamma_{int}, x_{int,0}),$$

where U_H and Y_L are the input sets, Y_H and U_L are the output sets, and $X_{int} = U_H$ is the state set. The initial state $x_{int,0} \in X_{int}$ can be chosen arbitrarily since, according to Def. 3.4.1, $0 \in \mathcal{T}$ and hence, the right-hand sides of (3.11) and (3.13) do not depend on $x_{int}(0)$. The transition function

$$\begin{aligned}\gamma^{int}(x(k), u_H(k), y_L(k)) = \gamma_x^{int}(x_{int}(k), u_H(k), y_L(k)) \times \\ \times \gamma_y^{int}(x_{int}(k), u_H(k), y_L(k)) \times \gamma_u^{int}(x_{int}(k), u_H(k), y_L(k))\end{aligned}$$

is defined as follows

$$x_{int}(k+1) = \gamma_x^{int}(x_{int}(k), u_H(k)) = \begin{cases} u_H(k), & k \in \mathcal{T} \\ x_{int}(k), & \text{otherwise.} \end{cases} \tag{3.11}$$

$$y_H(k) = \gamma_y^{int}(y_L(k)) = \begin{cases} \phi(y_L(k)), & k \in \mathcal{T} \\ \epsilon, & \text{otherwise,} \end{cases} \tag{3.12}$$

$$u_L(k) = \gamma_u^{int}(x_{int}(k), u_H(k)) = \begin{cases} u_H(k), & k \in \mathcal{T} \\ x_{int}(k), & \text{otherwise,} \end{cases} \tag{3.13}$$

where $\phi : Y_L \to Y_H$ is an A/D map as described in Sec. 3.4.1.

Here, we use the shorthand notation $k \in \mathcal{T}$ to denote the fact that an event is triggered at time k. We do not distinguish between the time-driven and event-driven dynamics since this does not influence the properties of the state machine. Recall that both triggering algorithms define a dynamic time scale. At time k, the state of the interface layer as well as the values of signals u_L and y_H change. At all remaining time instants both the state and the low-level signal u_L do not change. The value of the high-level output signal y_H is equal to ϵ, where ϵ is used to denote the "empty" symbol, i.e., the absence of an output symbol. Note that the use of the symbol ϵ allows us to project the high-level signal on the low-level time axis. In this way, the non-conflictingness condition can be checked without performing the transformation betwen two time axes.

Non-conflictingness of the hierarchical control system consisting of $\mathcal{B}_{pl}^L$, $\mathcal{B}_{int}$, and $\mathcal{B}_{sup}^H$ can be checked using methods described in Sec. 3.3. However, there is a specific feature. The plant $\mathcal{B}_{pl}$ and the interface layer $\mathcal{B}_{int}$ are synchronised w.r.t. the fast time scale $\mathbb{T}_f = \mathbb{N}_0$. At the same time, the interface layer $\mathcal{B}_{int}$ (and, hence, the composite system $\mathcal{B}_{int}^H[\mathcal{B}_{pl}^L]$) is synchronised with the high-level supervisor $\mathcal{B}_{sup}^H$ w.r.t. the different, "slow" time scale[3] $\mathbb{T}_{sl} = \mathbb{N}_0$ which is defined by $T(y_L)$. Therefore, while analysing structural properties of the interface layer we need to check the non-anticipation property w.r.t. (y_L, u_L) for all $k \in \mathbb{T}_f$, whereas the non-anticipation property w.r.t. (y_L, y_H), (u_H, u_L), and (u_H, y_H) have to be checked only at $k \in \mathcal{T}$.

Thus, we can see that the state machine P_{int} possesses the following properties: y_H does strictly not anticipate u_H and does not anticipate y_L; u_L does strictly not anticipate y_L and does not anticipate u_H.

[3] we use different notations to stress the fact that these time scales do not coincide.

3.5 Reconfigurable controller

Usually, the designer has a big set of "control instruments" that can be applied to the particular plant and the intermediate layer can be seen as a realisation of this "tool kit".

We consider two possible setups for a reconfigurable controller. Most reconfigurable controllers can be obtained as modifications of these two schemes.

3.5.1 Multi-controller

A general structure is shown in Fig. 3.4. We assume that the intermediate layer consists of the "bank" of local controllers and a switching mechanism controlled by the high-level input signal. The state evolution of these controllers is not continuous, they are even not required to be of the same dimension. The only condition that must be fulfilled is that the dimensions of input/output signal spaces must be equal for all controllers.

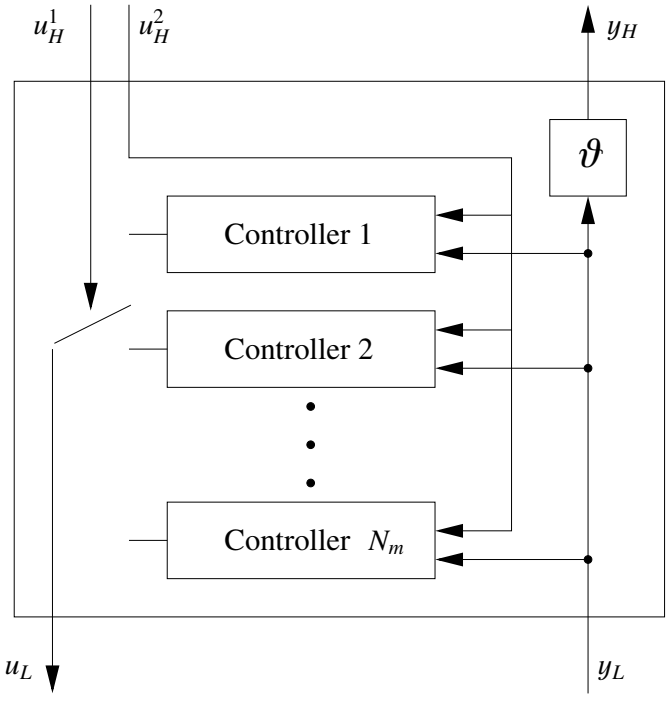

Figure 3.4: Reconfigurable controller with jumps in the state

Let the intermediate layer consist of N_m local controllers with dimensions of state space equal to $n_i,\ i = 1, \ldots, N_m$. Then the dynamics of the system can be described in the following way:

$$
\begin{aligned}
x_i(k+1) &= f_i(x_i(k), y_L(k), u_H^2(k)),\ i = 1, N_m \\
u_L(k) &= h_{u_H^1(k)}(x(k)),
\end{aligned}
$$

where $x_i(k) \in X_i \subseteq \mathbb{R}^{n_i}$ is the state vector of the i-th controller, $x = [x_1^T, \ldots, x_{N_m}^T]^T$ is the state of the whole system, $x(k) \in X \subset \mathbb{R}^N$, $N = \sum_{i=1}^{N_m} n_i$; $f_i : X_i \times Y_L \times U_H^2 \to X_i$ is the vector function determining the evolution of the corresponding controller, and $h_i : X_i \to U_L$ is the output function of the i-th controller. All N_m controllers work simultaneously. However, at each time instant k only one of them is connected to the plant input u_L. The active controller is determined by the high-level discrete control $u_H^1(k) \in U_H^1 = \{1, \ldots, N_m\}$. Finally, $\vartheta : Y_L \to Y_H$ is an aggregation function.

This kind of intermediate layer can be described by an I/S/- machine

$$
P_{mc} = ((U_H \times Y_L), (Y_H \times U_L), X_{mc}, \gamma_{mc}, X_{mc,0}),
$$

where $U_H = (U_H^1 \times U_H^2)$ and Y_L are the input sets, Y_H and U_L are the output sets, and $X_{mc} \subset \mathbb{R}^N$ is the state set. $X_{mc,0} \subset X_{mc}$ is the set of initial states and the transition function

$$
\begin{aligned}
\gamma^{mc}(x_{mc}(k), u_H(k), y_L(k)) = \gamma_x^{mc}(x_{mc}(k), u_H(k), y_L(k)) \times \\
\times \gamma_y^{mc}(x_{mc}(k), u_H(k), y_L(k)) \times \gamma_u^{mc}(x_{mc}(k), u_H(k), y_L(k))
\end{aligned}
$$

is defined as follows

$$
\begin{aligned}
x_{mc}(k+1) &= \gamma_x(x_{mc}(k), u_H(k), y_L(k)) &&= f(x_{mc}(k), y_L(k), u_H^2(k)), \\
y_H(k) &= \gamma_y(y_L(k)) &&= \vartheta(y_L(k)), \\
u_L(k) &= \gamma_u(x_{mc}(k), u_H(k)) &&= h_{u_H^1(k)}(x_{mc}(k)).
\end{aligned}
$$

Hence, we may observe that the state machine representing a multi-controller has the following properties: y_H does strictly not anticipate u_H and does not anticipate y_L; u_L does strictly not anticipate y_L and does not anticipate u_H.

An advantage of the multi-controller scheme is that it allows to combine rather different local control laws. For instance, we may implement an optimal control law for large transitions and a stabilisation law for the small neighbourhood of the equilibrium point. At the same time, this structure may have a number of drawbacks such as the saturation of the off-the-loop state and control signals as well as discontinuities in the control signal [84].

It has also been shown [84] that in many cases the bank of controllers can be realised by a single higher-order controller with a set of parameters. Each particular controller is chosen by assigning certain values to these parameters. This structure is called a reconfigurable controller with state sharing.

3.5.2 Reconfigurable controller with state sharing

The structure of a reconfigurable controller with continuous state evolution is presented in Fig. 3.5. The dynamics of the controller can be described by the following equations:

$$\begin{aligned} x(k+1) &= f_{u_H^1(k)}(x(k), y_L(k), u_H^2(k)) \\ u_L(k) &= h(x(k)), \end{aligned}$$

where $x(k) \in X \subseteq \mathbb{R}^n$, $u_H^1(k) \in U_H^1 = \{1, \ldots, N_m\}$ is the discrete control determining the current mode of operation for the continuous controller, $f_i : X \times Y_L \times U_H^2 \to X$, $i \in U_H^1$ are the corresponding right-hand sides of the controller equations, N_m is the number of modes, and ϑ is an aggregation function. In this setup we assume that the output function h does not depend on the mode of controller. This is natural, since we do not change the structure of the controller, but only its dynamics.

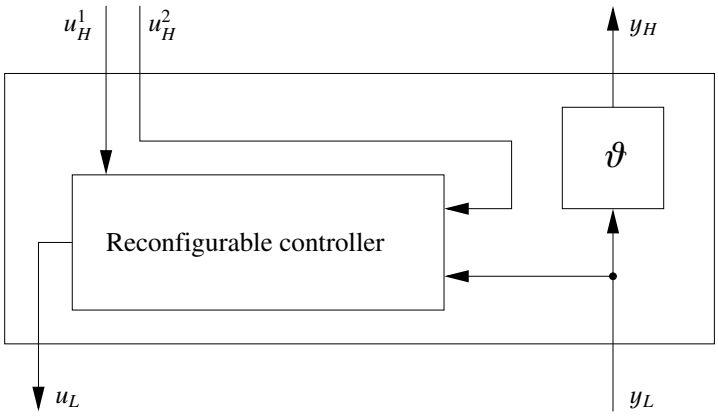

Figure 3.5: Reconfigurable controller with continuous state evolution

A reconfigurable controller with continuous state evolution can be described by an I/S/- machine

$$P_{rc} = ((U_H \times Y_L), (Y_H \times U_L), X_{rc}, \gamma_{rc}, X_{rc,0}),$$

where $U_H = (U_H^1 \times U_H^2)$ and Y_L are the input sets, Y_H and U_L are the output sets, and X_{rc} is the state set. $X_{rc,0}$ is the set of initial states and the transition function

$$\begin{aligned} \gamma^{rc}(x(k), u_H(k), y_L(k)) = \gamma_x^{rc}(x(k), u_H(k), y_L(k)) \times \\ \times \gamma_y^{rc}(x(k), u_H(k), y_L(k)) \times \gamma_u^{rc}(x(k), u_H(k), y_L(k)) \end{aligned}$$

is defined as follows

$$\begin{aligned} x(k+1) &= \gamma_x(x(k), u_H(k), y_L(k)) &&= f_{u_H^1(k)}(x(k), y_L(k), u_H^2(k)), \\ y_H(k) &= \gamma_y(y_L(k)) &&= \vartheta(y_L(k)), \\ u_L(k) &= \gamma_u(x(k)) &&= h(x(k)). \end{aligned}$$

It is easy to see that the state machine P_{rc} representing a reconfigurable controller has slightly different properties compared to P_{mc}: y_H does strictly not anticipate u_H and does not anticipate y_L; u_L does strictly not anticipate y_L and does *strictly* not anticipate u_H.

3.6 Multi-level hierarchy: non-conflictingness

In most cases one has to combine at least two intermediate control layers to implement the specification $\mathcal{B}_{sp}^{HL}$. Often it is advantageous to use several intermediate layers since it can simplify the design procedure. One layer can be responsible for the low-level control of the plant, whereas the second one can establish a communication between the low-level controller(s) and the high-level supervisor. On the other hand, as was shown in Sec. 3.2.2, an intermediate layer can be used to shape the behaviour of the plant and hence to simplify the implementation of the specification $\mathcal{B}_{sp}^{HL}$. Therefore, we need to extend our results to the case of multi-level hierarchical control structure.

In this section, we give a definition of a multi-level hierarchical control structure. Further, we formulate non-conflictingness conditions for the case of several intermediate layers and show how they can be proved in the most efficient way.

Let $W_i = U_i \times Y_i$ denote the signal space for the signals between the $(i-1)$-th and the i-th levels, $i = 1, N_\ell$. Σ_{pl}^0 is the plant with external behaviour $\mathcal{B}_{pl}^0 \subseteq W_1^{\mathbb{N}_0}$, Σ_{im}^i, $i = 1, N_\ell{-}1$ are the intermediate layers with external behaviours $\mathcal{B}_{im}^i \subseteq (W_i \times W_{i+1})^{\mathbb{N}_0}$, and $\Sigma_{sup}^{N_\ell}$ is the high-level supervisor with external behaviour $\mathcal{B}_{sup}^{N_\ell} \subseteq W_{N_\ell}^{\mathbb{N}_0}$. $\mathcal{B}_{pl}^0$ and $\mathcal{B}_{sup}^{N_\ell}$ are assumed to be I/- w.r.t. (U_1, Y_1) and (Y_{N_ℓ}, U_{N_ℓ}), respectively. The intermediate systems Σ_{im}^i are assumed to have the structure of a quadripole.

The composite low-level plants obtained through the recursive composition of the plant Σ_{pl}^0 with intermediate layers are denoted by Σ_{pl}^i, $i = 1, N_\ell{-}1$. The index i means that the plant Σ_{pl} has been composed with intermediate controllers up to the i-th level inclusive. The external behaviour $\mathcal{B}_{pl}^i \subset W_{i+1}^{\mathbb{T}}$ of the system Σ_{pl}^i is defined as follows:

$$\mathcal{B}_{pl}^i = \{w_{i+1} \in W_{i+1}^{\mathbb{T}} | \exists w_i \in \mathcal{B}_{pl}^{i-1} : (w_i, w_{i+1}) \in \mathcal{B}_{im}^i\},\ i = 1, N_\ell{-}1.$$

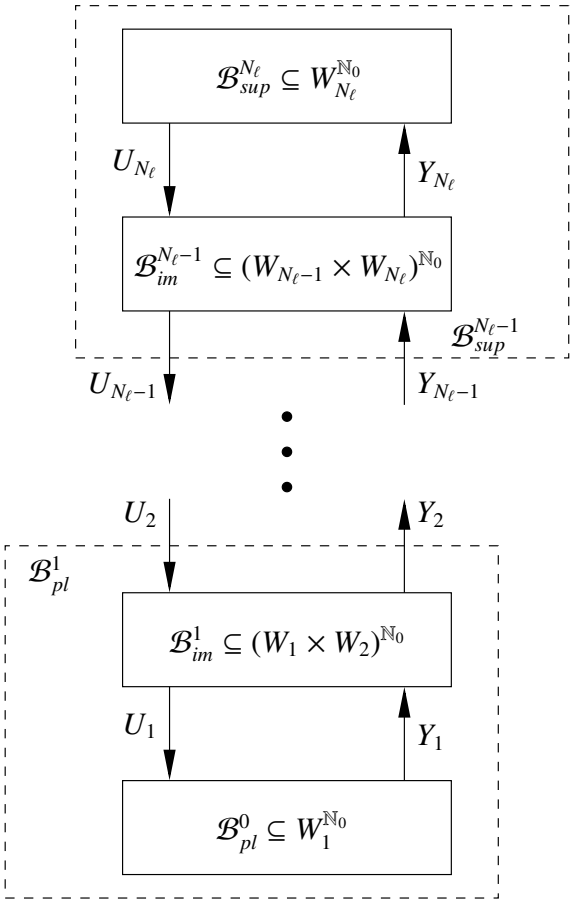

Figure 3.6: Multi-level hierarchy

In the same way, we can define the composite supervisors Σ^i_{sup}, $i = 1, N_\ell - 1$ with external behaviours $\mathcal{B}^i_{sup} \subset W_i^{\mathbb{T}}$:

$$\mathcal{B}^i_{sup} = \{w_i \in W_i^{\mathbb{T}} | \exists w_{i+1} \in \mathcal{B}^{i+1}_{sup} : (w_i, w_{i+1}) \in \mathcal{B}^i_{im}\},\ i = 1, N_\ell - 1.$$

Definition 3.6.1 *The tuple $(\mathcal{B}^1_{im}, \ldots, \mathcal{B}^{N_\ell-1}_{im}, \mathcal{B}^{N_\ell}_{sup})_{ml}$ is said to be an N_ℓ-level hierarchical solution to the control problem $(\mathcal{B}^0_{pl}, \mathcal{B}^0_{sp})_{cp}$ if*

i. $\mathcal{B}^0_{pl} \cap \mathcal{B}^1_{sup} \subseteq \mathcal{B}^0_{sp}$,

ii. $\mathcal{B}^i_{pl}$ *and* $\mathcal{B}^{i+1}_{im}$ *are non-conflicting for all* $i = 0, N_\ell - 2$,

iii. $\mathcal{B}^i_{sup}$ *and* $\mathcal{B}^{i-1}_{im}$ *are non-conflicting for all* $i = 2, N_\ell$,

iv. $\mathcal{B}^i_{pl}$ *is I/- w.r.t.* (U_{i+1}, Y_{i+1}), $\mathcal{B}^i_{sup}$ *is I/- w.r.t.* (Y_i, U_i) *and* $\mathcal{B}^i_{pl}$ *and* $\mathcal{B}^{i+1}_{sup}$ *are non-conflicting for all* $i = 0, N_\ell - 1$.

Requirements *ii.–iv.* of Def. 3.6.1 guarantee that all composition operations are well defined and there is no blocking at any level of the hierarchy. To ensure this one has to check $(3N_\ell - 2)$ conditions, which is very elaborate. The following theorem show that the non-conflictingness property of the multi-level hierarchical control system can be proved in N_ℓ steps.

Theorem 3.6.2 *To check non-conflictingness of the N_ℓ-level hierarchical solution to the control problem $(\mathcal{B}^0_{pl}, \mathcal{B}^0_{sp})_{\rm ml}$, it suffices to check the following N_ℓ conditions:*

i. $\mathcal{B}^i_{pl}$ is I/- w.r.t. (U_{i+1}, Y_{i+1}) and $\mathcal{B}^i_{pl}$ and $\mathcal{B}^{i+1}_{im}$ are non-conflicting for all $i = 0, N_\ell - 2$,

ii. $\mathcal{B}^{N_\ell-1}_{pl}$ is I/- w.r.t. (U_{N_ℓ}, Y_{N_ℓ}) and $\mathcal{B}^{N_\ell-1}_{pl}$ and $\mathcal{B}^{N_\ell}_{sup}$ are non-conflicting.

Equivalently, one can check an alternative set of N_ℓ conditions:

i. $\mathcal{B}^i_{sup}$ is I/- w.r.t. (Y_i, U_i) and $\mathcal{B}^i_{sup}$ and $\mathcal{B}^{i-1}_{im}$ are non-conflicting for all $i = 2, N_\ell$,

ii. $\mathcal{B}^1_{sup}$ is I/- w.r.t. (Y_1, U_1) and $\mathcal{B}^0_{pl}$ and $\mathcal{B}^1_{sup}$ are non-conflicting.

Proof. *We prove the first statement. The second one can be proved in the same way. To shorten notation we will write $\mathcal{B}_1 \bowtie \mathcal{B}_2$ to denote the fact that $\mathcal{B}_1$ and $\mathcal{B}_2$ are non-conflicting and that both $\mathcal{B}_1$ and $\mathcal{B}_2$ are I/- with respect to corresponding input/output signals.*

Hence, we need to prove that the following implication holds:

$$\left\{\begin{array}{rcl} \mathcal{B}^i_{pl} & \bowtie & \mathcal{B}^{i+1}_{im}, \quad i = 0, \ldots, N_\ell - 2 \\ \mathcal{B}^{N_\ell-1}_{pl} & \bowtie & \mathcal{B}^{N_\ell}_{sup} \end{array}\right. \Rightarrow \left\{\begin{array}{rcl} \mathcal{B}^j_{pl} & \bowtie & \mathcal{B}^{j+1}_{sup}, \quad j = 0, \ldots, N_\ell - 2 \\ \mathcal{B}^k_{im} & \bowtie & \mathcal{B}^{k+1}_{sup}, \quad k = 1, N_\ell - 1 \end{array}\right.$$

1. *Consider the case $j = N_\ell - 2$, $k = N_\ell - 1$ and the two-level hierarchy $(\mathcal{B}^{N_\ell-2}_{pl}, \mathcal{B}^{N_\ell-1}_{im}, \mathcal{B}^{N_\ell}_{sup})$. We have the following relations: $\mathcal{B}^{N_\ell-2}_{pl} \bowtie \mathcal{B}^{N_\ell-1}_{im}$ and $\mathcal{B}^{N_\ell-1}_{pl} \bowtie \mathcal{B}^{N_\ell}_{sup}$. Hence, we can apply Theorem 3.3.3 whence*

$$\mathcal{B}^{N_\ell}_{sup} \bowtie \mathcal{B}^{N_\ell-1}_{im}, \qquad (*)$$

$$\mathcal{B}^{N_\ell-1}_{sup} \bowtie \mathcal{B}^{N_\ell-2}_{pl}. \qquad (**)$$

2. *From Theorem 3.3.3 it follows that $\mathcal{B}^{N_\ell-1}_{sup}$ exists and is I/-. Hence, we can consider the two-level hierarchy $(\mathcal{B}^{N_\ell-3}_{pl}, \mathcal{B}^{N_\ell-2}_{im}, \mathcal{B}^{N_\ell-1}_{sup})$. By definition, we have $\mathcal{B}^{N_\ell-3}_{pl} \bowtie \mathcal{B}^{N_\ell-2}_{im}$, $(**)$ gives the second condition. Therefore, we can apply Theorem 3.3.3 again to obtain the next pair of relations.*

3. *This procedure can be iteratively repeated until we reach the lowest level.*

Herewith we have proved the above implications.

Note that the first set of conditions is more natural since the conditions are checked bottom-up which coincides with design procedure. Moreover, if there is an interface layer in the hierarchical control structure, the time aggregation introduces a natural ordering of the control layers, namely from those with the "fine" time scale to those with the "coarse" time scale.

Chapter 4

Optimal Control of Hybrid Automata under Safety and Liveness Constraints

When designing control for complex systems one often needs to fulfil requirements of a different nature. One typical problem statement for such systems – especially in safety related applications – is to minimise a cost function while respecting safety and liveness constraints. There are a number of abstraction-based control synthesis approaches that address safety and liveness issues while largely ignoring performance optimisation aspects [27, 103, 98, 78]. On the other hand, many interesting papers on the optimal control of hybrid systems have been presented [23, 48, 36, 107, 105, 7], but the proposed approaches are not able to handle "hard" safety constraints.

In this chapter we propose a hierarchical control structure for a low-level plant Σ_{pl} modeled by a discrete-time hybrid automaton (DTHA) $\mathcal{A}$. This structure consists of two levels as shown in Fig. 4.1. The first level, Σ_{DS}, contains an approximation based discrete supervisor which guarantees the fulfilment of safety and liveness constraints. We show that the action of this supervisor can be interpreted as restricting invariants of the hybrid automaton plant model. After the transformation, the composite system $\tilde{\Sigma}_{pl} = \Sigma_{DS}[\Sigma_{pl}]$ can be represented as a discrete-time hybrid automaton $\mathcal{A}_m$ with a modified set of invariants. This approach is based on the qualitative behaviour shaping technique described in Sec. 3.2.2.

The second level, Σ_{OC}, uses the remaining degrees of freedom to realise a state feedback control strategy which minimises a given cost function. This optimisation technique is based on the results on optimal control and stabilisation of switched systems [26] and hybrid automata [23, 24]. In these publications a method was presented to solve an infinite time horizon optimal

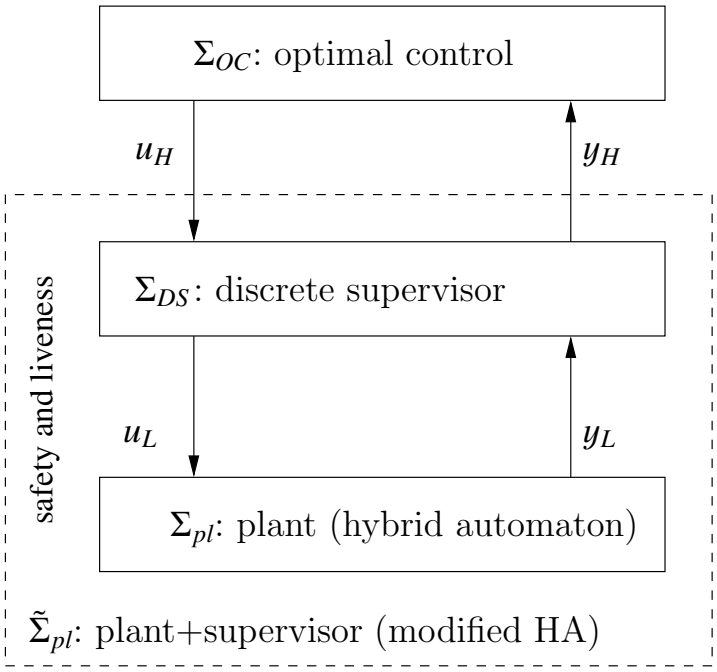

Figure 4.1: A hierarchical control architecture

control problem for a hybrid automaton with affine continuous dynamics, when a quadratic performance index is considered. These results have been generalised in order to cope with the particular structure of the plant:

(a) the existence of forbidden regions has been taken into account, assuming that the invariant sets inv_i may be proper subsets of the state space X, i.e., $inv_i \subsetneq X$;

(b) an infinite number of switches is allowed.

The resulting state feedback solution is defined on the basis of an appropriate state space partitioning. We show that this partition, which we call a *switching table*, can be efficiently computed off-line.

All results presented in this chapter were obtained in cooperation with the Automatic Control Group at the Department of Electrical and Electronic Engineering of the University of Cagliari, Italy. The fruitful collaboration with Daniele Corona, Carla Seatzu, and Alessandro Giua is gratefully acknowledged. The results have been published in conference proceedings [25, 43] and as a journal paper [106]. The presentation in this chapter is essentially based on [106].

The chapter is structured as follows. In Sec. 4.1, we recall some basic facts on hybrid automata, introduce the plant model and formalise the

specifications. In Sec. 4.2, the safety and liveness requirements are addressed using ℓ-complete abstraction of the continuous plant dynamics. In Sec. 4.3, the remaining degrees of freedom are used to minimise a quadratic cost function. A numerical example is provided in Sec. 4.4.

4.1 Plant Model and Specifications

In this section we first define the class of Hybrid Automata (HA) on which we focus in the following. Then we formally describe the safety specifications and the optimal control problem.

4.1.1 Hybrid Automata

Like a continuous-time hybrid automaton [5, 70], a discrete-time hybrid automaton $\mathcal{A}$ consists of a finite automaton extended with a continuous state. The latter, denoted by $x_L(k) \in \mathbb{R}^n$, evolves in discrete time $k \in \mathbb{N}_0$. Note that there is one difference compared to the "standard" definition of hybrid automata. Since an additional degree of freedom consists in the choice of locations, we interpret them as (restricted) discrete inputs. The hybrid automaton considered here is a structure

$$\mathcal{A} = (U_L, Y_L, X_L, f, q, inv, E),$$

where

- $U_L = \{u_L^1, \ldots, u_L^\alpha\}$ is a finite set of discrete inputs (locations);
- $Y_L \subseteq \mathbb{R}^m$ is an output space;
- $X_L \subseteq \mathbb{R}^n$ is a continuous state space;
- $f_i : X_L \to X_L$ is the transition function associated with every input symbol $u_L^i \in U_L$, i.e.

  $$x_L(k+1) = f_i(x_L(k)) \tag{4.1}$$

  if $u_L(k) = u_L^i$;
- $q : X_L \to Y_L$ is an output function;
- $inv : U_L \to 2^{X_L}$ is a function that associates an invariant $inv_i \subseteq X_L$ to each location $u_L^i \in U_L$.
- $E \subset U_L \times 2^{X_L} \times U_L$ is a set of edges. An edge $e_{i,j} = (u_L^i, g_{ij}, u_L^j) \in E$ is an arc between locations u_L^i and u_L^j with associated guard region g_{ij}. The set of edges describes restrictions on the sequence of control

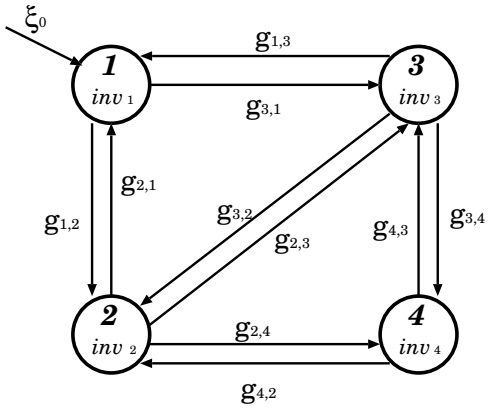

Figure 4.2: A graph describing an hybrid automaton.

symbols and can be interpreted as the discrete part of the overall hybrid automaton (see Fig. 4.2). This is a directed graph with vertices corresponding to the locations. The graph is assumed to be connected.

Starting from initial state $x_L(0) = x_0 \in X_L$ with input $u_L(0) = u_L^i$ such that $x_0 \in inv_i$, the continuous state x_L may evolve according to the corresponding discrete-time transition function f_i, i.e., $x_L(k+1) = f_i(x_L(k))$, until a switch to another location u_L^j occurs. The new location u_L^j has to satisfy the guard constraint $x_L(k) \in g_{ij}$ and $x_L(k) \in inv_j$. The future evolution of the continuous state is now determined by the transition function f_j, provided that the condition $f_j(x_L(k)) \in inv_j$ holds. A switch is enforced if the continuous state is about to leave the invariant inv_i, i.e. $f_i(x_L(k)) \notin inv_i$, $k \in \mathbb{N}_0$. However, the hybrid automaton may also switch to a "new" location u_L^j *before* being forced to leave its "old" location u_L^i, if the corresponding guard constraint and "new" invariant are satisfied. Thus, the sequence $u_L(k)$ of discrete locations can be interpreted as a *constrained control input*.

It may also happen that for some reachable state ($x_L(k)$) the system evolution cannot be extended to the interval $[k+1, \infty)$. This situation is referred to as *blocking*. The notion of *liveness*, in turn, corresponds to the fact that the system evolution can always be extended to infinity. Here, the liveness property of interest for the considered system is assumed to be equivalent to nonblocking. A first goal of low-level control is to assure this property along with safety specifications.

4.1.2 The Plant Model

We assume that the uncontrolled plant is modeled as a specific discrete-time hybrid automaton satisfying the following assumptions:

A4. The difference equation (4.1) is linear in x_L and time-invariant, i.e.

$$x_L(k+1) = A_i\, x_L(k) \quad \forall u_L^i \in U_L, k \in \mathbb{N}_0; \tag{4.2}$$

A5. $inv_i = X_L \quad \forall u_L^i \in U_L$;

A6. Transitions between any two locations $u_L^i, u_L^j \in U_L$ are allowed;

A7. $g_{ij} = X_L \quad \forall u_L^i, u_L^j \in U_L$;

A8. The state vector x_L is assumed to be trivially observable: $Y_L = X_L$, $q = \mathrm{Id}$ and, hence, $y_L(k) = x_L(k)\ \forall k \in \mathbb{N}_0$.

Note that assumption **A6** does not reduce the generality of the approach, since possible restrictions on discrete input signals can be considered as specifications – this will be illustrated in Sec. 4.2.2. Under the assumptions **A4 - A7**, the uncontrolled plant is a switched linear system with a *free* input signal $u_L : \mathbb{N}_0 \to U_L$.

Hence, the induced full behaviour of the discrete-time hybrid automaton $\mathcal{A}$ is $\mathcal{B}_{pl,f}^L \subseteq (U_L \times Y_L \times X_L)^{\mathbb{N}_0}$. We can see that the system $\Sigma_{pl} = (\mathbb{N}_0, U_L, Y_L, X_L, \mathcal{B}_{pl,f}^L)$ is an I/S/- system: input $u_L \in \mathcal{P}_{U_L}\mathcal{B}_{pl,f}^L = U_L^{\mathbb{N}_0}$ is locally free, the continuous state variable x_L satisfies the axiom of state, and, finally, u_L is strictly not anticipated by x_L and not anticipated by y_L.

The discrete-time hybrid automaton $\mathcal{A}$ under the assumptions **A4**–**A8** can therefore be represented as an I/S/- machine: $\Sigma_{pl} \simeq P_{pl}$. Note that the assumptions **A5**–**A7** are crucial and cannot be omitted. Moreover, since the function q does not depend on u_L, the state machine P_{pl} is strictly non-anticipating.

4.1.3 Specifications and problem decomposition

The overall control problem can be formulated in the following way: first, we define a set $\mathcal{B}_{sp}^L \subseteq (U_L \times Y_L)^{\mathbb{N}_0}$ of "admissible" I/O signals. We will call $\mathcal{B}_{sp}^L$ *safety* specifications.

To formalise the safety specifications, the continuous plant output space Y_L is partitioned w.r.t. the function $q_d : Y_L \to Y_d$, where Y_d is a finite set of symbols. To express dynamic safety constraints, certain sequences of input/output symbols are declared illegal or, in other words, the evolution of the hybrid automaton needs to be restricted such that only legal (U_L, Y_d)-sequences are generated. It is assumed that this set of sequences can be

realised by a finite state machine P_{sp}. The procedure of "building" such an automaton is described in detail in Sec. 4.2.2.

In a second step, subject to plant model and safety constraints, we aim at minimising the cost function

$$J = \sum_{k=0}^{\infty} y_L(k)' Q_{u_L(k)} y_L(k), \quad (u_L, y_L) \in \mathcal{B}^L_{pl} \cap \mathcal{B}^L_{sp}, \tag{4.3}$$

where, for each $k \in \mathbb{N}_0$ and $u_L(k) \in U_L$, $Q_{u_L(k)}$ is a positive semidefinite real matrix.

This problem will now be approached using a two-level control hierarchy. To address the safety issue, we define the specification $\mathcal{B}^{HL}_{sp}$ similar to the one proposed in Sec. 3.2.2. The low-level controller must obey this specification when connected to the plant. Moreover, the composite system low-level controller plus plant must be non-blocking (i.e., must satisfy liveness requirements). This is described in Section 4.2. The remaining degrees of freedom are used to minimise the cost function (4.3). This is described in Sec. 4.3.

4.2 The low-level task

In Sec. 3.2 we argued that operations with realisations for plant and low-level controller are extremely computationally expensive. However, the computational burden can be substantially reduced if both plant and specification can be described (or overapproximated) by finite-dimensional models. We show that the hybrid plant automaton can be efficiently approximated by a finite state machine employing the ℓ-complete approximation approach [98, 78]. Subsequently, Ramadge and Wonham's supervisory control theory [99] is used to synthesise a least restrictive supervisor. Note that, in general, controller synthesis and approximation refinement are iterated until a nontrivial supervisor guaranteeing liveness and safety for the approximation can be computed or computational resources are exhausted. In the former case, attaching the resulting supervisor to the hybrid plant model amounts to introducing restricted invariants. The resulting hybrid automaton represents the plant under low-level control and can be guaranteed to respect both safety and liveness constraints.

4.2.1 Ordered set of discrete abstractions

To formulate the problem within a discrete framework we need to discretise the continuous output signal y_L. For this purpose, we introduce the discretised output mapping $q_d : Y_L \to Y_d$:

$$y_d(k) = q_d(y_L(k)), \tag{4.4}$$

where the set of output symbols, Y_d, is assumed to be finite: $Y_d = \{y_d^1, \ldots, y_d^\beta\}$. Without loss of generality, the discretised output mapping q_d is supposed to be surjective (*onto*). It partitions the output space into a set of disjoint subsets $Y^{(i)} \subset Y_L$, $i = 1, \ldots, \beta$, i.e.

$$\bigcup_{i=1}^{\beta} Y^{(i)} = Y_L,$$
$$Y^{(i)} \cap Y^{(j)} = \emptyset \quad \forall i \neq j\,.$$

Hence, low-level control deals with a continuous system (4.2) with discrete external signals. $u_L : \mathbb{N}_0 \to U_L$ is the discrete control input and $y_d : \mathbb{N}_0 \to Y_d$ the discrete measurement signal.

To implement supervisory control theory, the hybrid plant model is approximated by a purely discrete one. This is done using the method of ℓ-complete approximation [98, 78], which is described in the following paragraphs.

Denote the external behaviour of the hybrid plant model by $\mathcal{B}_{pl}^L$, i.e. $\mathcal{B}_{pl}^L \subseteq (U_L \times Y_d)^{\mathbb{N}_0}$ is the set of all pairs of (discrete valued) input/output signals $w = (u_L, y_d)$ that (4.2) and (4.4) admit. Recall that $\mathcal{B}_{pl}^L$ is time-invariant. The crucial question here is whether $\mathcal{B}_{pl}^L$ is ℓ-complete. We know that for ℓ-complete systems we can decide whether a signal belongs to the system behaviour by looking at intervals of length ℓ. An ℓ-complete system can be represented by a difference equation in its external variables with lag ℓ. Hence, an ℓ-complete system can be realised by a *finite* state machine. However, the hybrid plant model $\mathcal{B}_{pl}^L$ is, except for trivial cases, not ℓ-complete. For such systems, the notion of *strongest ℓ-complete approximation* has been introduced in [78]: a time-invariant dynamical system with behaviour $\mathcal{B}_\ell$ is called strongest ℓ-complete approximation for $\mathcal{B}_{pl}^L$ if

$$\begin{aligned}
&(i) \quad && \mathcal{B}_\ell \supseteq \mathcal{B}_{pl}^L,\\
&(ii) \quad && \mathcal{B}_\ell \text{ is } \ell\text{-complete},\\
&(iii) \quad && \mathcal{B}_\ell \subseteq \tilde{\mathcal{B}}_\ell \text{ for any other } \ell\text{-complete } \tilde{\mathcal{B}}_\ell \supseteq \mathcal{B}_{pl}^L,
\end{aligned}$$

i.e. if it is the "smallest" ℓ-complete behaviour containing $\mathcal{B}_{pl}^L$. Obviously, $\mathcal{B}_\ell \supseteq \mathcal{B}_{\ell+1}$ $\forall \ell \in \mathbb{N}$, hence the proposed approximation procedure may generate an ordered set of abstractions. Clearly, $w \in \mathcal{B}_\ell \Leftrightarrow w|_{[0,\ell]} \in \mathcal{B}_{pl}^L|_{[0,\ell]}$. For $w|_{[0,\ell]} = (u_L^{i_0}, \ldots, u_L^{i_\ell}, y_d^{k_0}, \ldots, y_d^{k_\ell})$, where $i_j \in \{1, \ldots, \alpha\}$, and $k_j \in \{1, \ldots, \beta\}$ this is equivalent to

$$\begin{aligned}
&f_{i_{\ell-1}}\left(\ldots f_{i_1}\left(f_{i_0}\left(q_d^{-1}(y_d^{k_0})\right) \cap \left(q_d^{-1}(y_d^{k_1})\right)\right)\ldots\left(q_d^{-1}(y_d^{k_{\ell-1}})\right)\right) \cap q_d^{-1}(y_d^{k_\ell})\\
&\qquad := X(w|_{[0,\ell]}) \neq \emptyset.
\end{aligned} \tag{4.5}$$

Note that for a given string $w|_{[0,\ell]}$, $X(w|_{[0,\ell]})$ represents the set of possible values for the continuous state variable[1] $x_L(\ell)$ if the system has responded to the input string $u_L(0) = u_L^{i_0}, \ldots, u_L(\ell-1) = u_L^{i_{\ell-1}}$ with the output $y_d(0) = y_d^{k_0}, \ldots, y_d(\ell) = y_d^{k_\ell}$. Note also that (4.5) does not depend on $u_L(\ell)$. For linear and affine systems evolving in discrete time $\mathbb{N}_0$, (4.5) can be checked *exactly*, since all involved sets are polyhedra.

As both input and output signal evolve on finite sets U_L and Y_d, $\mathcal{B}_\ell$ can be realised by a (nondeterministic) finite automaton. In [98, 78], a particularly intuitive realisation is suggested, where the approximation state variable stores information on past values of u_L and y_d. More precisely, the automaton state set can be defined as

$$X_d := \bigcup_{j=0}^{\ell-1} X_{d_j},\ \ell \geq 1,$$

where $X_{d_0} = Y_d$, and X_{d_j} is the set of all strings $(u_L^{i_0}, \ldots, u_L^{i_{j-1}}, y_d^{k_0}, \ldots, y_d^{k_j})$ such that

$$(u_L^{i_0}, \ldots, u_L^{i_j}, y_d^{k_0}, \ldots, y_d^{k_j}) \in \mathcal{B}|_{[0,j]}.$$

The temporal evolution of the automaton can be illustrated as follows: From initial state $x_d(0) \in X_{d0}$, it evolves through states

$$x_d(j) \in X_{dj},\ 1 \leq j \leq \ell - 1$$

while

$$x_d(j) \in X_{d_{\ell-1}},\ j \geq \ell - 1.$$

Hence, until time $\ell-1$, the approximation automaton state is a complete record of the system's past and present, while from then onwards, it contains only information on the "recent" past and present.

As the states $x_d^i \in X_d$ of the approximation realisation are strings of input and output symbols, we can associate x_d^i with a set of continuous states, $X(x_d^i)$, in the same way as in (4.5).

Note that we can associate y_d^r as the unique output for each discrete state $x_d(r) = (u_L^{i_{r-j}}, \ldots, u_L^{i_{r-1}}, y_d^{k_{r-j}}, \ldots, y_d^{k_r}) \in X_d,\ j < \ell$. Thus, the output is just the last symbol in the symbolic description of the state. It is then a straightforward exercise to provide a transition function $\gamma_x : X_d \times U_L \to 2^{X_d}$ such that the resulting (non-deterministic) Moore-automaton $M_\ell = (U_L, Y_d, X_d, \gamma_x, \gamma_y, X_{d_0})$ with input set U_L, output set Y_d, state set X_d, output function $\gamma_y : X_d \to Y_d$, and initial state set X_{d_0} is a realisation of $\mathcal{B}_\ell$. Note that the state of M_ℓ is instantly deducible from observed variables [96].

[1] recall that the state x_L is trivially observable and, hence, we can find $q^{-1}(q_d^{-1}(y_d^i)) = \mathrm{Id}(q_d^{-1}(y_d^i)) \subset X_d$ for any $y_d^i \in Y_d$.

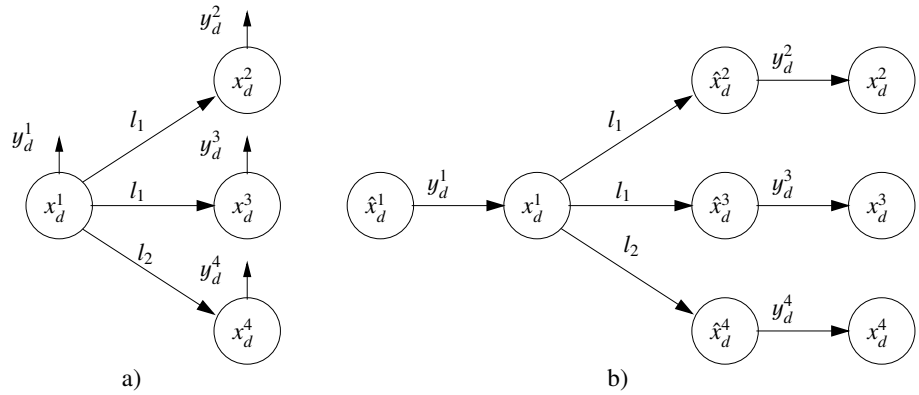

Figure 4.3: Moore-automaton a) and an equivalent automaton without outputs b). Note that $y_d^i = \gamma_y(x_d^i) \in Y_d$ is the output symbol associated with the discrete state x_d^i.

To recover the framework of supervisory control theory [99] as closely as possible, we finally convert M_ℓ into an equivalent finite state machine without input/output structure: $G_\ell = (\Sigma, \tilde{X}_d, \tilde{\gamma}_x, \tilde{X}_{d_0})$, where $\Sigma = U_L \cup Y_d$ represents the state set, U_L represents the set of controllable events and Y_d the set of uncontrollable events.

Technically, this procedure is carried out according to the following scheme (for an illustration, see Fig.4.3):

- Each state $x_d^j \in X_d$ is split into two states: x_d^j and $\hat{x}_d^j$. Thus, the new state set is formed as $\tilde{X}_d = X_d \bigcup \hat{X}_d$. Initial states are replaced by their complements, $\tilde{X}_{d_0} = \hat{X}_{d_0}$.

- The new transition function $\tilde{\gamma}_x$ is defined as a union of two transition functions with nonintersecting domains:

$$\tilde{\gamma}_x(\tilde{x}_d^i, \sigma^j) = \begin{cases} \backsim \tilde{x}_d^i, & \tilde{x}_d^i \in \hat{X}_d,\ \sigma^j = \gamma_y(\backsim \tilde{x}_d^i) \in Y_d, \\ \backsim \gamma_x(\tilde{x}_d^i, \sigma^j), & \tilde{x}_d^i \in X_d,\ \sigma^j \in U_L, \\ \text{undefined}\ , & \text{otherwise}, \end{cases}$$

  where $\backsim$ denotes an operation of taking the complementary state, i.e. $\backsim \hat{x}_d^i := x_d^i$ and vice versa. Note that the first event always belongs to the set Y_d, and the following evolution consists of sequences where events from U_L and Y_d alternate.

Since the function γ_y is scalar-valued, the set of feasible events for each state $x \in \hat{X}_d$ contains only one element. Thus, any two states x and $\backsim x$ form a fixed pair, where the states $\hat{X}_d$ are in some sense fictitious and play an auxiliary role.

4.2.2 Specifications and supervisor design

Specification $\mathcal{B}_{sp}^{HL}$

The specification $\mathcal{B}_{sp}^{HL}$ is designed to restrict the plant behaviour $\mathcal{B}_{pl}^{L}$ to those signals which satisfy the safety requirements. However, we do not restrict the signal spaces. Hence, the respective low-level and high-level signal spaces are equal: $U_L = U_H$, $Y_L = Y_H$. The goal of the intermediate level (low-level controller) is to restrict the control signals u_H in such a way that the resulting output signals y_H satisfy the safety constraints.

The specification $\mathcal{B}_{sp}^{HL}$ can be formally defined as follows:

$$\mathcal{B}_{sp}^{HL} = \left\{ (u_L, y_L, u_H, y_H) \in (U_L \times Y_L \times U_H \times Y_H)^{\mathbb{N}_0} \;\middle|\; \begin{array}{l} \mathcal{P}_{U_L \times Y_L} \mathcal{B}_{sp}^{HL} = \mathcal{B}_{sp}^{L} \\ u_H \in U_H^{\mathbb{N}_0}, \\ y_H = y_L, \end{array} \right\}$$

where $\mathcal{B}_{sp}^{L}$ is the safety specification. It is assumed that the specification $\mathcal{B}_{sp}^{L}$ can be realised by a finite automaton as shown below.

Realisations for specifications

In the following, we consider specifications which consist of independent specifications for the input and the output, respectively. The overall specification can be realised by a state machine with inputs and outputs. However, since we have chosen to design the low-level controller within the framework of supervisory control theory [99], we will represent the safety specification as a finite state machine (automaton) without input/output structure.

The specification on outputs expresses both static constraints (through restricting the set of allowed outputs $Y_d^* \subset Y_d$) and dynamic constraints. Dynamic constraints are usually represented as a set of forbidden strings such as, e.g. strings where "the symbol y_d^j follows immediately upon the symbol y_d^i", or "the symbol y_d^i appears three times one after another without any other symbol in between". The set of all allowed sequences is then realised as a finite automaton $P_Y = (Y_d, X_Y, \gamma_Y, x_{Y0})$.

The specifications for inputs, in turn, reflect structural restrictions on the allowed sequence of input symbols. They can be realised by a finite automaton $P_U = (U_L, X_U, \gamma_U, x_{U0})$.

The last stage is the composition of input and output specifications to obtain the overall specification P_{sp}. Note that to be compatible with the approximation automaton G_ℓ, the overall specification has to have a special transition structure, namely, it must generate only sequences of events that consist of alternating symbols $y_d \in Y_d$ and $u_L \in U_L$, where the first symbol

must belong to the set Y_d (see Fig. 4.3.b)). Thus, the resulting specification automaton is obtained as an "ordered" product of P_U and P_Y:

$$P = P_Y \vee P_U = (P_Y \mid\mid P_U) \times \Omega = (Y_d \cup U_L, X_{sp}, \gamma_{sp}, x_{sp,0}), \tag{4.6}$$

where the automaton Ω is given by $(Y_d \cup U_L, \{0,1\}, \gamma_\omega, \{0\})$,

$$\gamma_\omega(x,\sigma) = \begin{cases} 1, & x = 0,\ \sigma \in Y_d, \\ 0, & x = 1,\ \sigma \in U_L, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

To characterise the resulting specification automata, we need the notion of *current-state observability* [cf. 17, 88]:

Definition 4.2.1 *A finite state machine $A = (\Sigma, Q, \phi)$ is said to be* current-state observable *if there exists a nonnegative integer K such that for every $i \geq K$, for any (unknown) initial state $q(0)$, and for any admissible sequence of events $\sigma(0)\ldots\sigma(i-1)$ the state $q(i)$ can be uniquely determined. The parameter K is referred to as the* index of observability.

In the following, we assume that the automata P_U and P_Y are current-state observable with indices of observability K_{P_U} and K_{P_Y}, respectively.

The notion of current-state observability can be extended to the overall specification automaton. Its index of observability $K_{P_{sp}}$ is given by

$$K_{P_{sp}} = \begin{cases} 2K_{P_Y} - 1, & K_{P_Y} > K_{P_U}, \\ 2K_{P_U}, & K_{P_Y} \leq K_{P_U}. \end{cases} \tag{4.7}$$

Furthermore, to stay within the time-invariant framework we have to restrict ourselves to specifications that can be realised by strongly current-state observable automata:

Definition 4.2.2 *A finite state machine $A = (\Sigma, Q, \phi)$ is said to be* strongly current-state observable *if it is current-state observable with observability index K and if for each state $q \in Q$ there exists another state $q' \in Q$ such that the state q can be reached from q' by a sequence of K events, i.e. $\forall q \in Q\ \exists q' \in Q, \text{s.t. } q = \phi(q', s), s \in \Sigma^*, |s| = K$, where Σ^* is the Kleene closure of Σ and ϕ the extension of the automaton transition function to strings in Σ^*.*

Note that each state of such an automaton can be deduced from the string consisting of the past K events, independently from the initial state.

Supervisor design

Given an approximating automaton G_ℓ and a deterministic specification automaton (4.6) with observability index $K_{P_{sp}}$, supervisory control theory checks, whether there exists a nonblocking supervisor and, if the answer is affirmative, provides a least restrictive supervisor SUP by "trimming" the product of G_ℓ and P_{sp}. Hence the state set of the supervisor, X_{SUP}, is a subset of $\tilde{X}_d \times X_{sp}$.

The functioning of the resulting supervisor is very simple. At time k it "receives" a measurement symbol which triggers a state transition. In its new state x^j_{sup}, it enables a subset $\Gamma(x^j_{sup}) \subseteq U_L$ and waits for the next feedback from the plant. As shown in [78], the supervisor will enforce the specifications not only for the approximation, but also for the underlying hybrid plant model.

In the following, we will be interested in the special case of *quasi-static* specifications. To explain this notion, let $\mathcal{P}_{\tilde{X}_d} : X_{SUP} \to \tilde{X}_d$ denote the projection of $X_{SUP} \subseteq \tilde{X}_d \times X_{sp}$ onto its first component. If $\mathcal{P}_{\tilde{X}_d}$ is injective, i.e. if

$$\mathcal{P}_{\tilde{X}_d}(x_1) = \mathcal{P}_{\tilde{X}_d}(x_2) \Rightarrow x_1 = x_2,\ \forall x_1, x_2 \in X_{SUP} \tag{4.8}$$

then the specification automaton is called *quasi-static with respect to the approximation automaton* G_ℓ.

Lemma 4.2.3 *P is quasi-static with respect to G_ℓ if*

$$2\ell - 1 \geq K_P. \tag{4.9}$$

Proof. *Let x_1 and x_2 be two states of the supervisor SUP. There are two cases:*

1. *$\mathcal{P}_{\tilde{X}_d}(x_1), \mathcal{P}_{\tilde{X}_d}(x_2) \in \tilde{X}_{d_j},\ 1 \leq j < \ell - 1$. Each element from $\tilde{X}_{d_j}$ stores a record of the complete past and present of y_d and u_L. Since the specification automaton is assumed to be deterministic, this record unambiguously determines the current state of the specification automaton. Thus, (4.8) holds.*

2. *$\mathcal{P}_{\tilde{X}_d}(x_1), \mathcal{P}_{\tilde{X}_d}(x_2) \in \tilde{X}_{d_{\ell-1}}$. In this case an element from $\tilde{X}_{d_{\ell-1}}$ contains information only on "recent" past values of y_d and l. Precisely speaking, it contains information about the last ℓ output symbols and the last $\ell - 1$ control symbols. Thus, the complete record has length of $2\ell - 1$ symbols, which is sufficient to unambiguously determine the current state of P_{sp}.*

4.2.3 Low-level controller

The low-level controller is built on the basis of the designed supervisor SUP. We need to extend its structure to include the high-level signals u_H and y_H. The low-level controller is represented by the state machine $P_{im} = ((U_H \times Y_L), (Y_H \times U_L), X_{im}, \gamma_{im}, X_{im,0})$, where U_H and Y_L are the input sets, Y_H and U_L are the output sets, and X_{im} is the state set. The set of initial states $X_{im,0} = \tilde{X}_{d0} \times x_{sp,0}$ and the transition function

$$\gamma^{in}(x_{im}(k), u_H(k), y_L(k)) = \gamma_x^{im}(x_{im}(k), u_H(k), y_L(k)) \times \\ \times \gamma_y^{im}(x_{im}(k), u_H(k), y_L(k)) \times \gamma_u^{im}(x_{im}(k), u_H(k), y_L(k)).$$

Here, the transition function $\gamma_x^{im}(x_{im}(k), y_L(k))$ is equivalent to the transition function of SUP and $X_{im} = X_{SUP}$. The output functions γ_y^{im} and γ_u^{im} are defined as follows:

$$y_H(k) = \gamma_y^{im}(y_L(k)) = y_L(k),$$

$$u_L(k) = \gamma_u^{im}(x_{im}(k), u_H(k)) = \begin{cases} u_H(k), & u_H(k) \in \Gamma(x_{im}(k)), \\ \Gamma(x_{im}(k)), & \text{otherwise.} \end{cases}$$

Hence, the state machine P_{im} possesses following properties: y_H does strictly not anticipate u_H and does not anticipate y_L; u_L does strictly not anticipate y_L and does not anticipate u_H. Moreover, according to Lemma 3.3.2 the composite system plant Σ_{pl} plus low-level controller Σ_{DS}, $\Sigma_{pl}[\Sigma_{DS}]$, is non-conflicting and can be realised by a strictly non-anticipating state machine.

4.2.4 Plant model under low-level control

For the case of quasi-static specifications, each supervisor state x_{sup}^i corresponds exactly to a state $\tilde{x}_d^i = \mathcal{P}_{\tilde{X}_d}(x_{sup}^i)$ of the approximating automaton, which, in turn, can be associated with a set $X(\tilde{x}_d^i) = X(\mathcal{P}_{\tilde{X}_d}(x_{sup}^i))$. It turns out that if the overall specification is strongly current-state observable, the composite system plant plus supervisor can be represented by a hybrid automaton with modified invariants.

For $k \geq \ell - 1$, attaching the discrete supervisor to the plant model is therefore equivalent to restricting the invariants for each location $u_L^j \in U_L$ according to

$$inv_j = \bigcup_{\substack{i, \text{ s.t. } u_L^j \in \Gamma(x_{sup}^i) \\ \mathcal{P}_{\tilde{X}_d}(x_{sup}^i) \in X_{d_{\ell-1}}}} X(\mathcal{P}_{\tilde{X}_d}(x_{sup}^i)) \bigcup f_i(X(\mathcal{P}_{\tilde{X}_d}(x_{sup}^i))). \quad (4.10)$$

Note that for the initial time segment, i.e. $k < \ell - 1$, (4.10) is more restrictive than the discrete supervisor computed in Sec.4.2.2.

Hence the action of supervisory control is to restrict the invariants from $inv_j = X$ to inv_j given by (4.10) and, accordingly, to restrict the guards from $g_{ij} = X$ to $g_{ij} = inv_i \cap inv_j$ where inv_i and inv_j are computed as given in (4.10).

The union of all invariants inv_j, $j = 1, \ldots, \alpha$, forms the refined state set that contains only safe points, i.e. points for which exists at least one sequence of control symbols such that the resulting behaviour satisfies the specifications.

The resulting hybrid automaton represents the plant model under low-level control. As control system synthesis has been based on an ℓ-complete approximation, it is guaranteed that the resulting hybrid automaton satisfies safety and liveness requirements. The remaining degrees of freedom in choosing $u_L(k)$ can be used in a high-level controller addressing performance issues.

4.3 The high-level task

The high-level task requires the solution of an optimal control problem of the form (4.3).

The aim of this section is that of showing in detail that a state feedback solution of (4.3) can be obtained by computing off-line appropriate partitions of the state space, that we call *switching regions*, extending to the case at hand previous results on the optimal control of switched systems [105], based on dynamic programming arguments. In particular, we present the following three main results.

- Firstly, we recall how one can extend the results of [105] to the case of HA with invariants in order to compute an optimal state feedback control law for the problem (4.3) when a finite number of switches N is allowed.
- Then, we show how the proposed approach can be extended to the case of an infinite number of allowed switches.
- Finally, we show how to deal with the case of hybrid systems whose dynamics f_i are all unstable.

For sake of simplicity we will deal with completely connected automata. These results can be extended to the case of generic automata using the same arguments as in [23], where continuous-time HA were taken into account.

4.3.1 The optimal control problem with a finite number of switches

Let us now consider an optimal control problem of the form:

$$\begin{cases} V_N^*(i_0, y_{H,0}) & := \min_{\mathcal{I},\mathcal{K}} \left\{ F(\mathcal{I},\mathcal{K}) := \sum_{k=0}^{\infty} y_H(k)' Q_{i(k)} y_H(k) \right\} \\ \text{s.t.} & x(k+1) = A_{i(k)} x(k) \\ & y_H(k) = x(k) \\ & u_H(k) = i_r \in U_H, \text{ for } k_r \le k < k_{r+1}, \quad r = 0, 1, \ldots, N \\ & x(k) \in inv_{i(k)}, \text{ for } k = 0, 1, \ldots, +\infty \\ & 0 = k_0 \le k_1 \le \ldots \le k_N < k_{N+1} = +\infty \end{cases} \tag{4.11}$$

where Q_i are positive semi-definite matrices, $y_H(0) = y_{H,0}$ is the high-level output at time $k = 0$ which is equal to the initial state x_0, $u_H(0) = i_0$ is the high-level input at $k = 0$, and $N < +\infty$ is the maximum number of allowed switches, that is given a priori. Note that since the state is trivially observable we will consider the system state as output.

In this optimisation problem there are two types of decision variables:

- $\mathcal{I} := \{i_1, \ldots, i_N\}$ is a finite sequence of high-level input symbols (locations);
- $\mathcal{K} := \{k_1, \ldots, k_N\}$ is a finite sequence of switching time indices.

Problem (4.11) is well posed provided that the following hypothesis are verified.

A9. The invariant sets inv_i, $i \in U_H$, guarantee the liveness of the HA.

Note that Assumption **A9.** is generally not easy to verify. Nevertheless, in the case at hand, its satisfaction is guaranteed a priori by the low-level task, namely by the procedure used to construct the invariant sets.

Moreover, to ensure a finite optimal cost for any $x_0 \in \mathbb{R}^n$ and any $i_0 \in U_H$ we assume the following:

A10. There exists at least one location $i \in U_H$ such that A_i is strictly Hurwitz and $inv_i = \mathbb{R}^n$.

Note that this condition is sufficient but usually not necessary to get a finite optimal cost.

In [105] it was shown that under the assumption that $inv_i = \mathbb{R}^n$ for all $i \in U_H$, the optimal control law for the optimisation problem (4.11) takes the form of a state-feedback, i.e., it is only necessary to look at the current system state in order to determine if a switch from linear dynamics $A_{i_{k-1}}$ to A_{i_k}, should occur.

For a given location $i \in U_H$ when r switches are still available, it is possible to construct a table $\mathcal{C}_r^i$ that partitions the state space $\mathbb{R}^n$ into α regions $\mathcal{R}_j$'s, $j = 1, \cdots, \alpha = |U_L|$. Whenever $i_{N-r} = i$ we use table $\mathcal{C}_r^i$ to determine if a switch should occur: as soon as the continuous state x reaches a point in the region $\mathcal{R}_j$ for a certain $j \in U_H \setminus \{i\}$ we will switch to location $i_{N-r+1} = j$; no switch will occur if the continuous system's state x belongs to $\mathcal{R}_i$.

In [105] it was constructively shown how the tables $\mathcal{C}_r^i$ can be computed off-line using a dynamic programming argument: first the tables $\mathcal{C}_1^i$ $(i \in U_H)$ for the last switch are determined, then, by induction the tables $\mathcal{C}_r^i$ can be computed once the tables $\mathcal{C}_{r-1}^i$ are known.

In order to provide a graphical representation of $\mathcal{C}_r^i$ we associate a different colour to each dynamics A_j, $j \in U_H$. The region $\mathcal{R}_j$ of $\mathcal{C}_r^i$ is represented according to the defined colour mapping.

Note that when $inv_i = \mathbb{R}^n$ for all $i \in U_H$, the regions $\mathcal{R}_j$'s are homogeneous, namely if $\xi \in \mathcal{R}_j$ then $\lambda\xi \in \mathcal{R}_j$ for all $\lambda \in \mathbb{R}$. This implies that they can be computed by simply discretising the unitary semisphere. Clearly, this is no more valid when $inv_i \subsetneq \mathbb{R}^n$ for some $i \in U_H$, as in our case where a discretisation of all state space is necessary.

To show how the procedure of [105] can be extended to the case we are considering here, let $\xi \in \mathbb{R}^n$ be a generic vector, and let $\mathcal{D}$ be an appropriate set of points in the portion of the state space, which define the state space discretisation grid considered. Moreover, given a discrete state $i \in U_H$ and a continuous state $\xi \in \mathbb{R}^n$, we define the set

$$succ(\xi) = \{j \in U_H \ \mid \ \xi \in inv_j\}$$

which denotes the indices associated to the locations whose invariant set includes ξ.

The procedure to compute the switching regions is based on dynamic programming. Let us denote $T_r(i, \xi)$ the optimal remaining cost when the current continuous state is ξ, the current dynamics is A_i and r switches are still available. Thus, when $r = 0$, i.e., when no more switch may occur, $T_0(i, y) = \xi' Z_i \xi$ if A_i is Hurwitz and the system trajectory starting in ξ and evolving with dynamics A_i until the origin is reached, always stays within inv_i. The matrix Z_i is the solution of the discrete Lyapunov equation $A_i' Z_i + Z_i A_i = -Q_i$. In all the other cases, $T_0(i, \xi) = +\infty$.

The optimal remaining cost $T_r(i, \xi)$ for $r = 1, \ldots, N$, is computed recursively starting from $r = 1$ towards increasing values of r. More precisely, we first choose a finite time horizon $k_{\max}$ that is large enough to approximate the infinite time horizon. Then, for any $r = 1, \ldots, N$, any $\xi \in \mathcal{D}$ and any $i \in U_H$, we compute the value of the cost to infinity when the initial state is

(i, ξ), r switches are still available, the generic dynamics A_i is used for the first k times sampling, then the system switches to dynamics A_j. This cost, that we denote $T(i, \xi, j, k, r)$, is the sum of the cost due to the evolution with dynamics A_i for k times sampling, plus the optimal remaining cost from the new state $(j, A_i^k \xi)$ reached after the switch when $r-1$ switches are still available, i.e.,

$$T(i, \xi, j, k, r) = \xi' \left(\sum_{h=0}^{k} (A_i^h)' Q_i (A_i^h) \right) \xi + T_{r-1}(j, A_i^k \xi). \tag{4.12}$$

The previous equation expresses the dynamic programming argument used to efficiently compute the optimal switching law. When r switches are still available, we consider a nominal trajectory starting from a discretisation grid point ξ and evaluate its cost assuming we remain in the current dynamic i for a time k; the cost remaining after the after the switch the switch is evaluated on the basis of the table previously constructed for the $r-1$ switch. Note that for a fixed value of i, ξ, j and r we only need to find the optimal value of k with a one-dimensional search.

Note that only those evolutions that do not violate the invariant constraints should be taken into account. This means that if $\zeta = A_i^k \xi$ is the generic continuous state reached from ξ evolving with dynamics A_i for k steps, an evolution with dynamics A_j, $j \in U_H$, should be considered if and only if $j \in succ(\zeta)$.

Finally, we define the switching tables as mappings $\mathcal{C}_r^i : \mathcal{D} \to U_H$, and the generic region $\mathcal{R}_j$ as

$$\mathcal{R}_j = \{\xi \in \mathcal{D} \mid \mathcal{C}_k^i(\xi) = j\}.$$

The procedure to compute the switching regions is briefly summarised in the following algorithm.

Algorithm 2. Tables construction

Input: $A_i \in \mathbb{R}^{n \times n}, Q_i \in \mathbb{R}^{n \times n}, inv_i \subseteq \mathbb{R}^n (i \in U_H), N, k_{\max}, \mathcal{D}$.

Output: $\mathcal{C}_r^i, r = 0, 1, \ldots, N, i \in U_H$.

Notation: $\bar{Q}_i(k) = \sum_{h=0}^{k} (A_i^h)' Q_i (A_i^h)$, $Z_i = \lim_{k \to \infty} \bar{Q}_i(k)$.

Initialisation:

$r = 0$ `% remaining switches`
for $i = 1 : \alpha$
  for all $\xi \in \mathcal{D}$
    do: `Cost assignment:`

$$T_0(i,\xi) = \begin{cases} \xi' Z_i \xi & \texttt{if } A_i \texttt{ is Hurwitz and the system trajectory} \\ & \texttt{starting in } \xi \texttt{ and evolving with dynamics} \\ & A_i \texttt{ until the origin is reached, always} \\ & \texttt{stays within } inv_i \\ +\infty & \texttt{otherwise} \end{cases}$$

end for
end for

Computation of switching regions:

for $r = 1 : N$
  for $i = 1 : \alpha$
    for all $\xi \in \mathcal{D}$
      Computation of the remaining cost:
      set: $k = 0$, $\Delta = \emptyset$
      while $k \leq k_{\max}$
        $\zeta = A_i^k \xi$
        if $\zeta \notin inv_i$
          for all $j \in succ(\zeta)$
            $T(i, y, j, k, r) = \xi' \bar{Q}_i(k)\xi + T_{r-1}(j,k)$,
            $\Delta = \Delta \cup \{(j,k)\}$
          end for
          $k = k_{\max} + 1$
        else
          for all $j \in succ(\zeta) \setminus \{i\}$
            $T(i, \xi, j, k, r) = \xi' \bar{Q}_i(k)\xi + T_{r-1}(j,k)$,
            $\Delta = \Delta \cup \{(j,k)\}$
          end for
          $k = k + 1$
        end if
      end while
      if $i \in succ(\xi)$ and A_i is Hurwitz
        $T(i, \xi, i, k_{\max}, r) = \xi' Z_i \xi$,
        $\Delta = \Delta \cup \{(i, k_{\max})\}$
      end if
      do: Cost assignment: $T_r(i,\xi) = \min_{(j,k)\in\Delta} T(i,\xi,j,k,r)$
      do: Colour assignment: $(j^*, k^*) = \arg\min_{(j,k)\in\Delta} T(i,\xi,j,k,r)$,

$$\mathcal{C}_r^i(\xi) = \begin{cases} j^* & \texttt{if } k^* = 0 \\ i & \texttt{otherwise} \end{cases}$$

    end for
  end for

end for

Algorithm 2 first computes the optimal remaining cost $T_0(i,\xi)$ when no more switches are available. This is done for any $i \in U_H$ and any $\xi \in \mathcal{D}$. Then, the switching tables are computed backwards, starting from that corresponding to one available switch, until that corresponding to N available switches ($r = 1 : N$). More precisely, for any r, any location i with dynamics A_i and any sampling point $\xi \in \mathcal{D}$, we compute the optimal remaining cost starting from (i,ξ) when r switches are available. In order to do this we compare all the costs that can be obtained starting from (i,ξ), evolving with A_i for k sampling instants, then switching to any location A_j, and up to then evolving with the optimal evolution from (j,ζ), $\zeta = A_j^k\xi$, when $r-1$ switches are available. Note that k may only take finite values, namely $k = 0, 1, \ldots, k_{\max}$. This clearly does not affect the validity of the solution provided that $k_{\max}$ is taken large enough. If the minimum cost is obtained for $k = k^* = 0$ and $j = j^*$, this means that when the current state is (i,ξ) and r switches are still available, the cost is minimised if we immediately switch to location A_j. Therefore, we assign the colour corresponding to A_j to the point ξ in the table $\mathcal{C}_r^i$. On the contrary, if $k^* > 0$ it means that if the state is (i,ξ) and r switches are available, it is convenient to continue evolving with dynamics A_i. Therefore we assign the colour corresponding to A_i to the point ξ in the table $\mathcal{C}_r^i$.

The computational cost of the presented algorithm is of the order $\mathcal{O}(q^n N s^2)$ where n is the dimension of the state space and q is the number of samples in each direction (i.e., q^n is the cardinality of $\mathcal{D}$). Therefore, the complexity is a quadratic function of the number of possible dynamics and linear in the number of switches.

Two important cautions should be taken in order to ensure the uniqueness of the tables and (as it will be discussed in the following) the absence of the Zeno behaviour when the procedure is extended to the case of $N = \infty$.

The argument (j^*, k^*) that minimises the cost $T(i,\xi,j,k,r)$ may be not unique and this may cause ambiguity in the construction of the tables. To this aim we introduce the following lexicographic ordering.

Let $\Gamma = \{(j,k) \in \Delta \mid (j,k) = \arg\min T(i,\xi,j,k)\}$ be the set of solutions of problem

$$T_r(i,\xi) = \min_{(j,k)\in\Delta} T(i,\xi,j,k,r), \tag{4.13}$$

and assume that Γ has cardinality greater than one. Let (j',k') and (j'',k'') be any two couples in Γ with $j' \neq j''$. We say that $(j',k') \prec (j'',k'')$ iff $j' < j''$. Finally, if $j' = j''$ we say that $(j',k') \prec (j'',k'')$ iff $k'' < k'$.

Choosing the minimal element in Γ with respect to $\prec$, the optimal solution of Problem (4.13) is unique.

The second precaution that should be taken is the following. Consider the case in which at a given value of the switching index k, the arguments that minimise the remaining cost $T(i, \xi, j, k)$ starting from point ξ in dynamics A_i are (j^*, k^*) with $k^* = 0$. It may be the case that the system, once entered in dynamics A_{j^*}, requires an immediate switch to another dynamics, say p causing the presence of 2 switches in zero time. This behaviour is undesirable, because it leads to a potential risk of a Zeno behaviour when the number of available switches goes to infinite.

To avoid this it is sufficient to take (p, k^*) instead of (j^*, k^*), or more precisely, to consider $(j^*, k^*) = \arg\min T(j^*, \xi, j, k)$ at the previous switching index $r - 1$. When this extra precaution is taken, we can ensure that a spacing condition $k_{r+1} - k_r > 0$ is always verified during an optimal evolution.

4.3.2 The optimal control problem with an infinite number of switches

In [26] it was shown that under the assumption that $inv_i = \mathbb{R}^n$ for all $i \in U_H$, the above procedure can be extended to the case of $N = \infty$, provided that (i) for at least one $i \in U_H$, A_i is Hurwitz, and (ii) for all $i \in U_H$, $Q_i > 0$.

Analogous results can be proved here under Assumptions **A9** and **A10**, and under the additional following hypothesis.

A11. For all $i \in U_H$, $Q_i > 0$.

Lemma 4.3.1 *For any continuous initial state χ_0, $\chi_0 \neq 0$, and $\forall\ \varepsilon > 0$, $\exists\ \bar{N}$ such that for all $N > \bar{N}$,*

$$\frac{V_N^*(i, \chi_0) - V_{\bar{N}}{}^*(j, \chi_0)}{V_N^*(i, \chi_0)} < \varepsilon,$$

for all $i, j \in U_H$.

Proof. *See [106].*

According to the above result, one may use a given fixed relative tolerance ε to approximate two cost values, i.e.,

$$\frac{V_N^*(i, \chi) - V_{N'}{}^*(j, \chi)}{V_N^*(i, \chi)} < \varepsilon \implies V_N^*(i, \chi) \cong V_{N'}{}^*(j, \chi).$$

This result enables us to prove the following important theorem.

Theorem 4.3.2 *Given a fixed relative tolerance ε, if $\bar{N}$ is chosen as in Lemma 4.3.1 then for all $i, j \in U_H$ it holds that $\mathcal{C}^i_{\bar{N}+1} = \mathcal{C}^j_{\bar{N}+1}$.*

Proof. *It trivially follows from the fact that, by Lemma 4.3.1,* $V^*_{\bar{N}+1}(i,\chi_0) = V^*_{\bar{N}+1}(j,\chi_0)$ *for all* $i,j \in U_H$*, and from the uniqueness of the optimal tables as discussed above.*

This result also allows one to conclude that

$$\forall\ i \in U_H, \quad \mathcal{C}_\infty = \lim_{N\to\infty} \mathcal{C}^i_N,$$

i.e., *all tables converge to the same one.*

To construct the table $\mathcal{C}_\infty$ the value of $\bar{N}$ is needed. We do not provide so far any analytical way to compute $\bar{N}$, therefore our approach consists in constructing tables until a convergence criterion is met.

Table $\mathcal{C}_\infty$ can be used to compute the optimal feedback control law that solves an optimal control problem of the form (4.11) with $N = \infty$. More precisely, when an infinite number of switches is available, we only need to keep track of the table $\mathcal{C}_\infty$. If the current continuous state is χ and the current location is A_i, on the basis of the knowledge of the colour of $\mathcal{C}_\infty$ in χ, we decide if it is better to still evolve with the current dynamics A_i or switch to a different dynamics, that is univocally determined by the colour of the table in χ.

Let us finally observe that table $\mathcal{C}_\infty$ is *Zeno-free*, i.e., it guarantees that no Zeno instability may occur when it is used to compute the optimal feedback control law. This property is guaranteed by the procedure used for their construction as discussed in Sec. 4.3.1.

4.3.3 The optimal control of switched systems with unstable dynamics

In this section we show that the above results still apply when Assumption **A10.** is relaxed. To this aim, let us first introduce the following definitions.

Definition 4.3.3 (Forbidden region) *A forbidden region for the* $\mathcal{A}$ *is a set* $X_f \subset X : X_f = X \setminus \bigcup_{i=1}^{s} inv_i$*, where* α *is the number of locations.*

Thus X_f is a region forbidden to *all* dynamics of the $\mathcal{A}$.

Definition 4.3.4 (Augmented $\mathcal{A}$ and OP) *An* augmented automaton $\overline{\mathcal{A}} = (\overline{U}_H, \overline{Y}_H, \overline{X}, \overline{f}, \overline{q}, \overline{inv}, \overline{E})$ *of* $\mathcal{A} = (U_H, Y_H, X, f, q, inv, E)$ *and the corresponding optimal control problem* $\overline{OP}$ *of* OP *(4.11), are related as follows:*

(i) $\overline{\mathcal{A}}$ *includes a new Hurwitz dynamics* $A_{\alpha+1}$ *and* $\overline{OP}$ *includes a corresponding weight matrix* $Q_{\alpha+1} = k\tilde{Q}_{\alpha+1}$ *(with* $rank(\tilde{Q}_{\alpha+1}) > 0$*, and* $k > 0$*).*

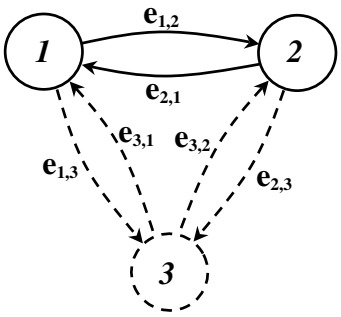

Figure 4.4: Graph of the automaton $\mathcal{A}$ (continuous) and $\overline{\mathcal{A}}$ (continuous and dashed) described in the example.

(ii) A new invariant $inv_{\alpha+1} = X$ *is associated to the new dynamics.*

(iii) The edges $e_{i,\alpha+1} \in \overline{E}$ *and* $e_{\alpha+1,i} \in \overline{E}$ *are defined* $\forall i \in \overline{U}_H$.

■

Thus the augmented automaton $\overline{\mathcal{A}}$ is the same as $\mathcal{A}$ except for an extra input symbol ($u_H^{\alpha+1}$) completely connected to all the locations in the $\mathcal{A}$. Its invariant set coincides with $inv_{\alpha+1} = X$ and its dynamics is $A_{\alpha+1}$. The corresponding $\overline{OP}$ weights location $(\alpha + 1)$ with matrix $Q_{\alpha+1} > 0$.

The following important result holds.

Lemma 4.3.5 *Assume that there exists an exponentially stabilising switching law for problem* $OP(\mathcal{A})$. *Then there also exists a sufficiently large value of* $k > 0$ *in the* $\overline{OP}(\overline{\mathcal{A}})$, *such that the table* $\overline{\mathcal{C}}_\infty$, *solution of* $\overline{OP}(\overline{\mathcal{A}})$, $i = 1, \ldots, \alpha + 1$, *contain the colour of* $A_{\alpha+1}$ *at most in* X_f.

Lemma 4.3.5 allows one to consider the solution of $\overline{OP}(\overline{\mathcal{A}})$ equivalent to the solution of $OP(\mathcal{A})$. This follows from the fact that the dynamics $A_{\alpha+1}$ does not influence at all any solution of the augmented problem. Therefore it can be removed from the augmented automaton.

This result is formally proved in [26], in absence of state space constraints. As before this result can be trivially extended if the liveness of the automaton is guaranteed (see Assumption **A9.**). In fact, by definition, it holds that, for any initial state $\chi_0 \in X \setminus X_f$ and input symbol i_0 of the $\mathcal{A}$, the trajectory $\chi(k)$ corresponding to the the solution of $OP(\mathcal{A})$, $i(k)$, always keeps within $X \setminus X_f$.

Let us finally observe that the convergence to a unique table $\overline{\mathcal{C}}_\infty$ is due to the fact that we are dealing with strongly connected HA. If such were not the case, then α different tables $\overline{\mathcal{C}}_\infty^i$, $i = 1, \ldots, \alpha$, would be obtained as

a solution of $\overline{OP}$, and all of them should be used to compute the optimal feedback control law.

4.3.4 Robustness of the procedure

The high-level procedure we suggest in this chapter provides a switching table, i.e., a partition of the state space, that the controller consults on-line in order to establish which is the current location that ensures the minimality of the cost function. If no disturbance is acting on the system and no numerical error affects the switching table construction, the optimality of the solution, as well as the safeness and liveness of the closed-loop system, are guaranteed.

In practice, two different problems may occur.

- The first one concerns disturbances acting on the continuous system that may change its nominal trajectory. If the disturbance does not bring the system inside the forbidden region X_f, this does not affect the validity of the result: the controller continues taking its decisions on the basis of the current continuous state, the minimality of the cost is guaranteed, and the safety and liveness constraints are still satisfied.

  On the contrary, the procedure obviously fails if the disturbance is such that the trajectory enters the forbidden region X_f: in this case the safety and liveness constraints cannot be satisfied any more.

- The second problem is related to the inevitable numerical errors that affect the construction of the switching tables due to the state space discretisation required by our approach. For each switch, the table is computed for all points that belong to the discretisation grid. From these points the nominal trajectories are studied piecewise using Eq. (4.12): after the switching the remaining evolution may start from a point that does not belong to the grid and the actual remaining trajectory is an approximation of the nominal one.

  The actual trajectories during the system evolution will be close to the nominal ones used to compute the tables if the discretisation step is sufficiently small. A numerical error in the computation of the tables is not critical if the forbidden region is empty. In such a case, the provided solution may be sub-optimal but it is still viable.

  However, assume that the forbidden region is not-empty. If the forbidden regions constraints the optimal solution, it is likely that an optimal nominal trajectory needs to pass as close as possible to the forbidden region without entering it. However, an evolution that differs from a

nominal one may graze or even hit the forbidden region. In this case, the safety and liveness constraints cannot be satisfied any more.

A solution we suggest to overcome both problems is the following.

In the low level task we define a "tolerance region" X_{tr} around the forbidden state space set X_d, then we redefine the forbidden state space set as $X'_d = X_d \cup X_{tr}$. This clearly implies a reduction of the invariant sets of each location ($inv'_1, \ldots, inv'_\alpha$), and consequently, a wider forbidden region $X'_f \supset X_f$ in the high level task.

The security region should be large enough to make sure that a trajectory that differs from a nominal one – either because a disturbance is acting on the system, or because of the numerical errors in the construction of the tables – may pass within X'_f but never reaches X_f.

In this way we improve the robustness of the procedure. We have to pay a cost for this: the optimality of the solution with respect to the chosen performance index is no more guaranteed, and the computed solution will be suboptimal.

4.4 Numerical example

Let us consider a $\mathcal{A}$ with two locations, whose graph is depicted in Figure 4.4 (the part sketched with continuous lines). Moreover, let

$$A_1 = \begin{bmatrix} 0.981 & 0.585 \\ -0.065 & 0.981 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0.981 & 0.065 \\ -0.585 & 0.981 \end{bmatrix}$$

be the corresponding continuous dynamics, whose eigenvalues have unitary norm. In particular, in both cases it holds $\lambda_{1,2} = 0.9808 \pm j0.1951$. Two generic trajectories relative to dynamics A_1 and A_2, respectively, are reported in Figure 4.5.

Assume that

$$X = \{x \in \mathbb{R}^2 \ \mid \ x_1^2 + 9x_2^2 \leq 40 \ \wedge \ 9x_1^2 + x_2^2 \leq 40\},$$

where $x_1^2 + 9x_2^2 \leq 40$ and $x_1^2 + x_2^2 \leq 40$ are the equations of the trajectories trough the point of coordinates $x_1 = x_2 = 2$ and evolving with dynamics A_1 and A_2, respectively.

Finally, assume that the safety constraint is given by the forbidden state space set

$$X_d = \{x \in \mathbb{R}^2 | H'x \leq h\}$$

where

$$\begin{aligned} H &= \begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \\ h &= \begin{bmatrix} 0.8 & -0.2 & 0 & 0 \end{bmatrix}, \end{aligned} \tag{4.14}$$

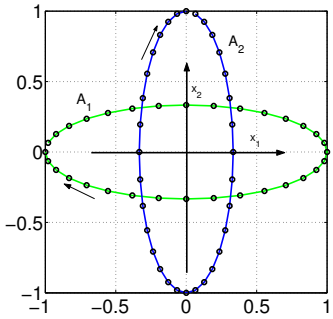

Figure 4.5: Discrete time trajectories of dynamics A_1 and A_2, with eigenvalues along the unitary circle.

i.e., X_d is the trapezoid depicted in Fig. 4.6.

The set $X \setminus X_d$ can be blocking, i.e., there exists some initial point in $X \setminus X_d$ such that, regardless of the switching strategy, any trajectory starting from these points always hit the set X_d.

In order to guarantee liveness, the previous setup is passed to the procedure described in Section 4.2. In terms of safety specification it means that we mark all output symbols $y_d \in \hat{Y}_d$, where $\hat{Y}_d = \{y_d \in Y_d \,|\, q_d^{-1}(y_d^{(i)}) \cap X_d \neq \emptyset\}$, as forbidden and require that the allowed sequences of output symbols do not contain such forbidden symbols. Then, the obtained supervisor is transformed into the appropriate invariant sets inv_1 and inv_2 to be associated to the dynamics A_1 and A_2, respectively. A new forbidden region $X_f = X \setminus (inv_1 \cup inv_2) \supset X_d$ is defined according to Definition 4.3.3.

The invariant sets of locations 1 and 2 are reported in Fig. 4.6.(a) and (b), respectively, while the set X_f is sketched in figure (c).

Within the given constraints we want to solve an optimal control problem[2] OP of the form (4.3), where $Q_1 = Q_2 = I$. For this purpose we consider the augmented problem $\overline{OP}(\overline{\mathcal{A}})$, with the following data:

$$A_3 = \begin{bmatrix} 0.9838 & 0 \\ 0 & 0.9838 \end{bmatrix}, \qquad Q_3 = kQ_1, \qquad inv_3 \equiv X$$

where $k = 10^3$, and A_3 is Hurwitz. The graph of the augmented automaton is depicted in Fig. 4.4 (continuous and dashed part).

Given the symmetry of the two dynamics, it can be easily shown that the solution of $OP(\mathcal{A})$ when $inv_i \equiv X$, $i = 1, 2$, is to use dynamics A_2 when $x_1 x_2 > 0$ and dynamics A_1 when $x_1 x_2 < 0$. This result is very intuitive if we observe the trajectories of the given dynamics (Figure 4.5) and if we use the identity matrices as weight matrices in problem (4.3).

[2]Note that neither A_1 nor A_2 are Hurwitz, hence an infinite number of switches is necessary.

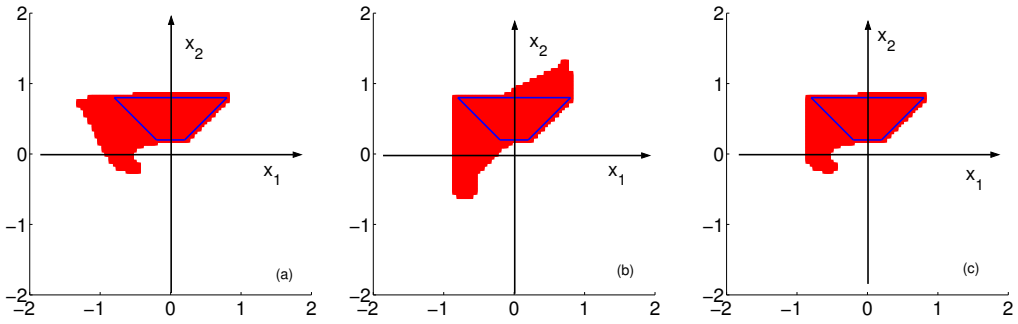

Figure 4.6: Invariants (in white) of locations 1 (a) and 2 (b) and (c) the forbidden region $X_f = X \setminus (inv_1 \bigcup inv_2)$ defined in Def. 4.3.3. The interior of the blue trapezoid is the forbidden region X_d.

Note that the augmented problem $\overline{OP}(\overline{\mathcal{A}})$ satisfies the conditions given in Definition 4.3.4. The switching table procedure, applied to $\overline{OP}(\overline{\mathcal{A}})$ for a recursively increasing number of switches, converges after $N = 15$ switches. Moreover the tables $\mathcal{C}^i_\infty$, $i = 1, 2, 3$ are coincident with a unique table $\mathcal{C}_\infty$, because the state graph of $\overline{HA}$ (Fig. 4.4) is strongly connected.

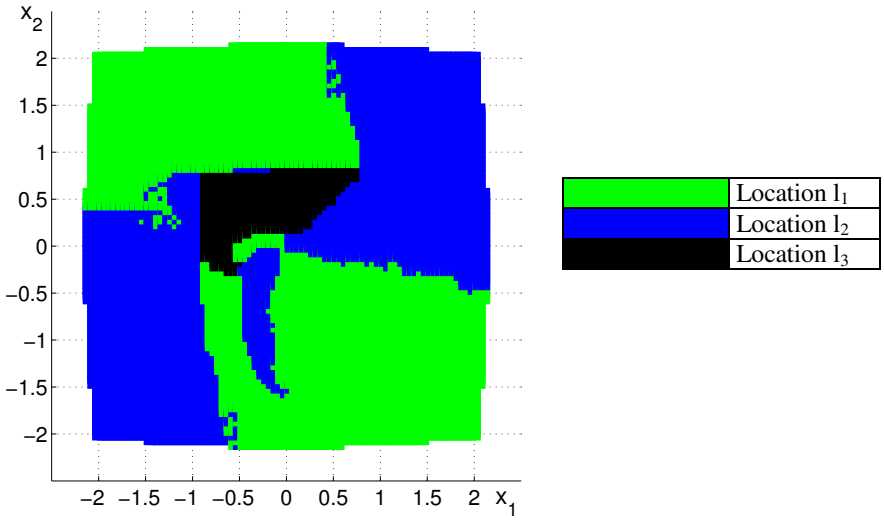

Figure 4.7: Switching table of the problem $\overline{OP}(\overline{\mathcal{A}})$ defined in the example.

The table $\mathcal{C}_\infty$ is depicted in Figure 4.7 and some relevant considerations can be immediately done.

(i) The colour of the augmented dynamics exactly covers the region X_f.

(ii) In $inv_1 \setminus X_f$ ($inv_2 \setminus X_f$) the col or is that relative to dynamics A_2 (A_1).

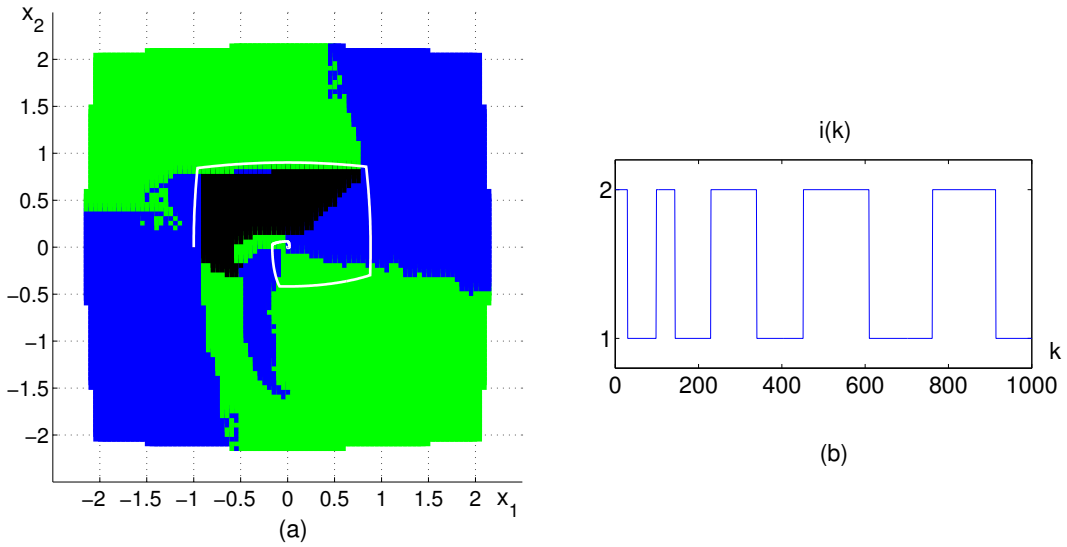

Figure 4.8: Trajectories $x(k)$ (a) and $i(k)$ (b) of the optimal solution of $OP(\mathcal{A})$ obtained by using the table in Figure 4.7 for an initial point ($i_0 = 1, x_0 = [-1\ 0]'$).

(iii) Around the origin the solution of $OP(HA)$ coincides with the solution described above relative to the case of $inv_i = X$ for $i = 1, 2$.

From the above table we deduce that there exists a finite optimal solution for any initial state $x_0 \in X \setminus X_f$ of the HA; moreover, if $x_0 \in X_f$ the optimal solution of $\overline{HA}$ uses dynamics A_3 for the minimum time required to leave X_f; from then on the optimal solution of HA is used. This can be viewed by the simulations depicted in Figure 4.8(a). The optimal cost from the point $x_0 = [-1\ 0]'$ with $i_0 = 1$ is $J = 196.6$. For completeness also the index trajectory $i(k)$ is reported in Figure 4.8(b).

The total computational time (Matlab 7, on an Intel Pentium 4 with 2 GHz and 256 Mb RAM) for constructing the table in Figure 4.7 is about 40 hours. This time is big, because a very dense space discretisation was considered (approximately 2×10^3 points). It is important, however, to point out that this computational effort is spent off-line. The on-line part of the procedure consists in measuring the state $x(k)$ and comparing its value with the switching table to decide the optimal strategy.

Chapter 5

Detecting and Enforcing Monotonicity for Hybrid Control Systems Synthesis

In the previous section, we presented an approach to hierarchical control synthesis where a finite state machine was used to "replace" the continuous plant dynamics. Different approaches to the discrete abstraction (or over-approximation) of continuous dynamics have been proposed during the last decade (see, e.g. [27, 58, 21, 65, 78]).

One key issue in designing such abstractions is to make sure that the behaviour of the abstraction covers (over-approximates) the behaviour of the continuous dynamics on a suitable (discrete) external signal space. This, in turn, boils down to computing guaranteed overapproximations for reachability sets in the continuous component's state space. For general nonlinear systems, this represents a highly nontrivial problem. However, if the system under consideration is monotone (e.g. [110, 2]) with respect to a partial order in its state space, an abstraction can be computed in a straightforward way [79].

Unfortunately, in most cases the systems under consideration are not monotone. Therefore, a question arises – whether it is possible to use low-level control to enforce monotonicity. In this chapter we formulate this problem within the framework developed in Sec. 3.2. We show how to check for the existence of a suitable partial order and how to design the low-level control to enforce monotonicity of the composite system plant plus low-level controller.

The results presented in this chapter have been published as a conference paper [42].

This chapter is organised as follows: in Section 5.1, we briefly review

the notion of a partial order. Section 5.2 addresses the concept of monotone, i.e., order preserving, dynamical systems, both for the autonomous and the controlled case. This section contains new results on how to efficiently check monotonicity (Lemmata 5.2.3 and 5.2.6). In Section 5.3, we briefly outline how monotonicity can be used in the context of abstraction based hybrid control synthesis. Finally, in Section 5.4.2, we formulate the problem within the hierarchical control framework and investigate how appropriate continuous feedback on a lower level of a hierarchical hybrid control scheme can enforce monotonicity and hence facilitate the computation of discrete abstractions for higher level control purposes.

5.1 Partial order relations

A partial order relation $\preceq$ on a Banach (or, more precisely, ordered metric) space B is defined as an operation satisfying the following three properties:

$$\begin{array}{lll} 1. & x \preceq x & \forall x \in X, \\ 2. & (x \preceq y) \ \wedge \ (y \preceq z) \Rightarrow x \preceq z & \forall x, y, z \in X, \\ 3. & (x \preceq y) \ \wedge \ (y \preceq x) \Rightarrow x = y & \forall x, y \in X. \end{array}$$

We write $x \prec y$ if $x \preceq y$ and $x \neq y$. This relation is no longer reflexive and is referred to as a strict order relation. Usually, to introduce an order relation one uses an auxiliary set $K \subset B$, such that

$$\begin{array}{lll} 1. & \alpha k \in K & \forall k \in K, \ \alpha \in \mathbb{R}_+, \\ 2. & k_1 + k_2 \in K & \forall k_1, k_2 \in K, \\ 3. & k \in K \wedge -k \in K \Rightarrow k = 0. & \end{array}$$

Thus, K is a convex pointed cone. Given K, we define the relation $x \preceq y$ if and only if $y - x \in K$.[1] If K has nonempty interior int$K$ then we define $x \prec\prec y$ iff $y - x \in \text{int}K$. It is stronger than $\prec$ or $\preceq$ as $x \prec\prec y$ implies $x \prec y$ and therefore $x \preceq y$. In Euclidean space $\mathbb{R}^n$, orthants can play the role of cones. Each orthant $\mathbb{R}^n_\delta \subset \mathbb{R}^n$ is characterised by its signature, i.e. the n-tuple $\delta = \{\delta_1, \ldots, \delta_n\}$ whose elements take values from the two-element set $\{0, 1\}$. $\mathbb{R}^n_\delta$ is defined as $\mathbb{R}^n_\delta = \{x \in \mathbb{R}^n | (-1)^{\delta_i} x_i \geq 0\}$. Hence, the zero signature corresponds to the positive orthant. We use notation $\preceq_\delta$ (resp., $\prec_\delta$ and $\prec\prec_\delta$) to show that the corresponding relation is defined with respect to the orthant $\mathbb{R}^n_\delta$. Relation symbols without index refer to relations w.r.t. the positive orthant.

[1] $x \prec y$ iff $y - x \in K \backslash \{0\}$

5.2 Monotone dynamical systems

A monotone dynamical system is a dynamical system on an ordered metric space which has the property that ordered states remain ordered when time progresses. In other words, monotone systems are order preserving dynamical systems. In this section we give some conditions for an arbitrary autonomous dynamical system to be monotone. Furthermore, these results are extended to dynamical systems with inputs.

5.2.1 Autonomous systems

Consider the dynamical system:

$$\dot{x}(t) = f(x(t)), \tag{5.1}$$

where $x(t) \in X \subset \mathbb{R}^n$, $f : X \to \mathbb{R}^n$ is a continuously differentiable vector field. The solution of (5.1) that starts at the point x_0 at $t = 0$ is defined as $\phi_t(x_0)$ and referred to as the flow of (5.1). To make an assertion about qualitative properties of the above dynamical system we have to introduce some classification.

Definition 5.2.1 *A vector field $f : X \to \mathbb{R}^n$ is said to be of type K_δ on an open subset $D \subset X$ if for each $i \in \{1, \ldots, n\}$, $(-1)^{\delta_i} f_i(a) \leq (-1)^{\delta_i} f_i(b)$ for any two points a and b in D satisfying $a \preceq_\delta b$ and $a_i = b_i$.*

The following lemma ([110], Chapt. 3, Prop. 5.1) asserts that the type K_δ condition is necessary and sufficient for the order preserving property to hold.

Lemma 5.2.2 *Let f be of type K_δ on D and $x_0, y_0 \in D$. If $x_0 \preceq_\delta y_0$ (resp., $x_0 \prec_\delta y_0$ or $x_0 \prec\prec_\delta y_0$), $t > 0$, and if $\phi_t(x_0)$ and $\phi_t(y_0)$ are defined and in D, then $\phi_t(x_0) \preceq_\delta \phi_t(y_0)$ (resp., $\phi_t(x_0) \prec_\delta \phi_t(y_0)$ or $\phi_t(x_0) \prec\prec_\delta \phi_t(y_0)$).*

The most natural way to decide whether a vector field f is of type K_δ is to analyse the sign structure of the Jacobian matrix of f. More specifically, it can be shown ([110]) that the vector field $f(x)$ is of type K_δ on the convex subset D if and only if

$$(-1)^{\delta_i + \delta_j} \frac{\partial f_i}{\partial x_j}(x) \geq 0, \quad i \neq j,\ x \in D. \tag{5.2}$$

Condition (5.2) can be checked in two steps:

Step 1: Check whether the off-diagonal elements of the Jacobian matrix are *sign-stable*, i.e.

$$\left\{ \frac{\partial f_i(x)}{\partial x_j} \geq 0\ \forall x \in D \right\} \wedge \left\{ \frac{\partial f_i(x)}{\partial x_j} \leq 0\ \forall x \in D \right\} \tag{5.3}$$

and *sign-symmetric*, i.e.

$$\frac{\partial f_i(x)}{\partial x_j} \cdot \frac{\partial f_j(x)}{\partial x_i} \geq 0 \; \forall x \in D \tag{5.4}$$

for all $i, j \in \{1, \ldots, n\}$ such that $i \neq j$.

Step 2: If the tests in Step 1 are satisfied, we need to check whether the (Boolean) equalities

$$\delta_i \oplus \delta_j = s_{ij}, \quad i < j \tag{5.5}$$

hold, where $\oplus$ represents "exclusive OR" and the $n(n-1)/2$ variables s_{ij}, $i < j,\ j = 2, \ldots, n$ are defined as follows:

$$s_{ij} = \begin{cases} 0 & \text{if } \dfrac{\partial f_i(x)}{\partial x_j} > 0 \vee \left(\dfrac{\partial f_i(x)}{\partial x_j} = 0 \wedge \dfrac{\partial f_j(x)}{\partial x_i} > 0 \right) \\ 1 & \text{if } \dfrac{\partial f_i(x)}{\partial x_j} < 0 \vee \left(\dfrac{\partial f_i(x)}{\partial x_j} = 0 \wedge \dfrac{\partial f_j(x)}{\partial x_i} < 0 \right) \\ \text{arbitrary in} \{0,1\} & \text{if } \dfrac{\partial f_i(x)}{\partial x_j} = \dfrac{\partial f_j(x)}{\partial x_i} = 0, \quad \forall x \in D. \end{cases} \tag{5.6}$$

Often, one wants to check whether a given vector field is of type K_δ for some (yet unknown) sign structure δ. Step 1 obviously remains the same, but in Step 2 we need to decide whether (5.5) is solvable for the unknown $\delta = \{\delta_1, \ldots, \delta_n\}$. The following proposition presents an easy way to do this. Moreover, it shows that if the answer is positive, the orthant signature can be easily extracted from the sign structure of the Jacobian matrix.

Lemma 5.2.3 *The system of Boolean equations (5.5) is solvable w.r.t. δ_i if and only if the following condition is satisfied:*

$$s_{ij} \oplus s_{ik} = s_{jk}, \quad i < j,\ j < k,\ i, j, k \leq n. \tag{5.7}$$

Proof. **(necessity)** . *Let us rewrite expression $s_{ij} \oplus s_{ik}$ using (5.5):*

$$s_{ij} \oplus s_{ik} = \delta_i \oplus \delta_j \oplus \delta_i \oplus \delta_k.$$

By definition, $a \oplus b \equiv b \oplus a$, $a \oplus a \equiv 0$ and $a \oplus 0 \equiv a$. Thus, $s_{ij} \oplus s_{ik} = \delta_j \oplus 0 \oplus \delta_k = \delta_j \oplus \delta_k = s_{jk}$, and we have shown that (5.7) follows from (5.5).

(sufficiency). *We now show that (5.7) implies that*

$$\delta = \{0, s_{12}, \ldots, s_{1n}\} \tag{5.8}$$

is a solution of (5.5). $\delta_1 \oplus \delta_j = 0 \oplus s_{1j} = s_{1j}$ holds trivially for $j = \{2, \ldots, n\}$, and $\delta_i \oplus \delta_j = s_{1i} \oplus s_{1j} = s_{ij}$, $i, j \in \{2, n\}, j > i$, where the last equality follows from (5.7).$\square$

Similarly, it can be shown that

$$\tilde{\delta} = \{1, s_{12} \oplus 1, \ldots, s_{1n} \oplus 1\} \tag{5.9}$$

is also a solution of (5.5) if (5.7) holds. Furthermore, (5.8) and (5.9) represent the only solutions. This can be shown by considering a vector δ' with $\delta'_i \neq \delta_i$ (i.e. $\delta'_i = \delta_i \oplus 1$) for some $i \in \{1, \ldots, n\}$ and $\delta'_j = \delta_j$ for some $j \neq i$. Hence,

$$\delta'_i \oplus \delta'_j = \delta_i \oplus 1 \oplus \delta_j = s_{ij} \oplus 1 \neq s_{ij},$$

which shows that δ' is not a solution of (5.5).

Note that (5.8) and (5.9) signify orthants that are symmetric w.r.t. the origin.

For linear systems

$$\dot{x}(t) = Ax(t),$$

the Jacobian matrix is the A matrix, i.e. $J(x, u) = A$. Thus, the sign structure of the Jacobian is completely determined by the signs of the elements a_{ij}. Obviously, they are sign-stable, so we need to check only conditions (5.4) and (5.5). Condition (5.4) (*sign-symmetry*) holds if $a_{ij}a_{ji} \geq 0$, $i \neq j$. The second step is to check the corresponding Boolean equation (5.5) using the method described in Lemma 5.2.3.

Example 5.2.4 *Let us consider the Jacobian matrix with the following sign structure*

$$J = \begin{bmatrix} * & + & 0 & - \\ + & * & + & 0 \\ 0 & + & * & 0 \\ - & 0 & 0 & * \end{bmatrix}.$$

Here we use asterisks to stress the fact that diagonal elements do not affect the monotonicity property. According to (5.6), we have $\{s_{12}, s_{14}, s_{23}\} = \{0, 1, 0\}$ *while* s_{13}, s_{24}, *and* s_{34} *are arbitrary. From Lemma 5.2.3 we can deduce that (5.5) is solvable iff* $\{s_{13}, s_{24}, s_{34}\} = \{0, 1, 1\}$. *The corresponding signature is* $\delta = \{0, 0, 0, 1\}$.

5.2.2 Controlled systems

Some of the previous results can be extended to dynamical systems driven by an exogenous input signal. A system

$$\dot{x}(t) = f(x(t), u(t)), \tag{5.10}$$

where $x(t) \in X \subset \mathbb{R}^n$, $u(t) \in U \subset \mathbb{R}^m$, $f : X \times U \to \mathbb{R}^n$, generates a flow $\phi_t(x_0, u_\tau)$, $u_\tau = u(\tau)$, $0 \leq \tau \leq t$, which represents a solution of (5.10) with initial condition $x(0) = x_0$ and external input signal u.

Definition 5.2.5 *A controlled dynamical system (5.10) is monotone w.r.t. the orthants $\mathbb{R}^n_\delta$ and $\mathbb{R}^m_\gamma$ if the following implication holds for all $t \geq 0$:*

$$x_1 \preceq_\delta x_2,\ u_1(\tau) \preceq_\gamma u_2(\tau),\ 0 \leq \tau \leq t \Rightarrow$$
$$\phi_t(x_1, u_{1\tau}) \preceq_\delta \phi_t(x_2, u_{2\tau}).$$

In [2], a condition for the controlled system (5.10) to be monotone w.r.t. the orthants $\mathbb{R}^n_\delta$ and $\mathbb{R}^m_\gamma$ has been proposed.

Proof. **([2])** *The system (5.10) is monotone w.r.t. the orthants $\mathbb{R}^n_\delta$ and $\mathbb{R}^m_\gamma$ if and only if the following properties hold for all $x \in D$ and all $u \in U$:*

$$(-1)^{\delta_i+\delta_j} \frac{\partial f_i}{\partial x_j}(x,u) \geq 0, \quad i \neq j,\ i,j \leq n$$

$$(-1)^{\delta_i+\gamma_j} \frac{\partial f_i}{\partial u_j}(x,u) \geq 0, \quad i \leq n,\ j \leq m.$$

The above conditions are, in fact, the extended variant of condition (5.2) from the previous section. Hence, in addition to conditions (5.3), (5.4) and (5.5), which are used to check (5.2), the following tests need to be performed:

First, the partial derivatives w.r.t. the control variables need to be *sign stable*, i.e.

$$\frac{\partial f_i(x,u)}{\partial u_j} \geq 0 \text{ or } \frac{\partial f_i(x,u)}{\partial u_j} \leq 0, \quad \forall x \in D, \forall u \in U \tag{5.11}$$

for all $i \leq n,\ j \leq m$. Moreover, the set of Boolean equations

$$\delta_i \oplus \gamma_j = q_{ij}, \quad i \leq n,\ j \leq m, \tag{5.12}$$

where

$$q_{ij} = \begin{cases} 0 & \text{if} \quad \dfrac{\partial f_i(x)}{\partial u_j} > 0, \\ 1 & \text{if} \quad \dfrac{\partial f_i(x)}{\partial u_j} < 0, \\ \text{arbitrary in } \{0,1\} & \text{if} \quad \dfrac{\partial f_i(x)}{\partial u_j} = 0, \end{cases} \tag{5.13}$$

needs to be solvable with respect to the vector $\gamma = \{\gamma_1, \ldots, \gamma_m\}$.

The following lemma gives a necessary and sufficient condition for equations (5.5) and (5.12) to be solvable.

Lemma 5.2.6 *The systems of Boolean equations (5.5) and (5.12) are solvable if and only if the following conditions are satisfied:*

$$s_{ij} \oplus s_{ik} = s_{jk}, \quad i < j,\ j < k,\ i, j, k \leq n, \tag{5.14}$$

$$q_{ij} \oplus q_{kj} = s_{ik}, \quad i \neq k,\ i, k \leq n,\ j \leq m. \tag{5.15}$$

Moreover,

$$\delta = \{0, s_{12}, \ldots, s_{1n}\},$$
$$\gamma = \{q_{11}, \ldots, q_{1m}\}$$

is a solution.

Proof. *Proof. The proof can be carried out according to the same scheme as in Lemma 5.2.3.*□

It can be shown that the solution is also defined up to inversion, i.e.

$$\tilde{\delta} = \{1, s_{12} \oplus 1, \ldots, s_{1n} \oplus 1\},$$
$$\tilde{\gamma} = \{q_{11} \oplus 1, \ldots, q_{1m} \oplus 1\}$$

is the only other solution of (5.5), (5.12).

Condition (5.15) can be represented as

$$q_{k1} \oplus q_{l1} = s_{kl}, \quad k < l,\ k, l \leq n,$$

$$col_1(Q) \stackrel{\oplus}{=} col_j(Q) \quad \forall i, j \leq m,$$

where $Q = q_{ij}, i \in \{1, \ldots, n\},\ j \in \{1, \ldots, m\}$, and $\stackrel{\oplus}{=}$ denotes an equality up to the inversion w.r.t. $\oplus$.

5.2.3 Special cases

In the following we point out two special cases of a controlled system (5.10) where the simpler Lemma 5.2.3 suffices to check monotonicity.

a) If $u(t)$ is entirely defined by the present state $x(t)$, i.e. $u(t) = u(x(t))$, the Jacobian of the closed loop system is

$$J_{ij}(x) = \frac{\partial f_i(x, u(x))}{\partial x_j} + \sum_{l=1}^{m} \frac{\partial f_i(x, u(x))}{\partial u_l} \frac{\partial u_l(x)}{\partial x_j}. \tag{5.16}$$

and the procedure described in Lemma 5.2.3 can be applied to (5.16).

b) In a hybrid control context, the control vector often consists of two components, i.e. $u'(t) = [u_1'(t),\ u_2'(t)]$, where $u_1(t) \in \mathbb{R}^k$, $k < m$ is determined by continuous state feedback, i.e. $u_1(t) = u_1(x(t))$, and u_2 is a piecewise constant signal with finite range $\mathcal{U} \subset \mathbb{R}^{m-k}$, $|\mathcal{U}| = N \in \mathbb{N}$. In this

case, the system can be treated separately on intervals, where u_2 is constant, i.e. $u_2(t) = u_\kappa \in \mathcal{U}$ $t \in [t_\kappa, t_{\kappa+1})$, and Lemma 5.2.3 can be applied again. The value u_κ is interpreted as a parameter, and the Jacobian is given by

$$J^\kappa_{ij}(x, u_\kappa) = \frac{\partial f_i(x, u_1(x), u_\kappa)}{\partial x_j} + \sum_{l=1}^{k} \frac{\partial f_i(x, u_1(x), u_\kappa)}{\partial u_{1l}} \frac{\partial u_{1l}(x)}{\partial x_j} \quad \forall \kappa \in \mathbb{N}.$$

Note that in our hybrid systems context, monotonicity is only needed to compute safe abstraction. Hence, having monotonicity w.r.t. different orthants for different values of κ will not pose any problems.

5.3 The role of monotonicity in abstraction based control synthesis

To demonstrate the usefulness of the monotonicity property in control synthesis, we will now briefly describe a specific hybrid system. Consider a continuous system

$$\dot{x}(t) = g_{u_2(t)}(x(t)) \tag{5.17}$$

where, as indicated before, u_2 is a piecewise constant signal with finite range $\mathcal{U}$, $|\mathcal{U}| = N$.

$$z(t) = h(x(t)) \tag{5.18}$$

is a discrete-valued output signal with finite range, i.e. $h : \mathbb{R}^n \to \mathcal{Z}$, $|\mathcal{Z}| = M < \infty$. Let us further assume that the system (5.17), (5.18) is sampled, either on a regular sampling grid ("time-driven sampling") or on the sampling grid defined by the output signal z ("event-driven sampling"). In the latter case, the input may only be switched at the time instances where the output changes. In both cases, Equations (5.17), (5.18) and the considered sampling device form a continuous system (with state set $X \subset \mathbb{R}^n$) evolving in discrete time $\mathbb{N}_0$ on a discrete external signal space $\mathcal{U} \times \mathcal{Z}$. Let $\mathcal{B} \subseteq (\mathcal{U} \times \mathcal{Z})^{\mathbb{N}_0}$ denote its behaviour. For abstraction based control synthesis, we need a discrete approximation, evolving on the same external signal space and exhibiting behaviour $\mathcal{B}_{\text{ab}} \supseteq \mathcal{B}$. In Sec. 4.2.1, strongest ℓ-complete approximation was advocated as a particularly suitable abstraction.

From a computational point of view, determining the strongest ℓ-complete approximation boils down to deciding whether a given string of input and output symbols $(u_2^{i_0}, \ldots, u_2^{i_\ell}, z^{k_0}, \ldots, z^{k_\ell})$ is an element in $\mathcal{B}|_{[0,\ell]}$. To obtain a precise answer, we would need to compute the evolution of the quantisation cell $h^{-1}(z^{k_0})$ under the flow $\phi_{u_2^{i_0}}$ associated with $g_{u_2^{i_0}}$, intersect the result with $h^{-1}(z^{k_1})$, track the evolution of the result under the flow $\phi_{u_2^{i_1}}$ associated with $g_{u_2^{i_1}}$ etc. To obtain safe approximation, or abstraction, it is

sufficient to compute outer approximations of the mappings of quantisation cells and their intersections. Clearly, if g_{u_2} is monotone w.r.t. the partial order $\preceq$, and quantisation cells are "boxes" w.r.t. $\preceq$, then $\phi_{u_2^i}(h^{-1}(z^k))$ is "trapped" within the the evolution of "external points", i.e. $a \preceq h^{-1}(z^k) \preceq b$ implies

$$\phi_{u_2^i}(a) \preceq \phi_{u_2^i}(h^{-1}(z^k)) \preceq \phi_{u_2^i}(b).$$

It is then a straightforward exercise to compute the required outer approximations and hence the desired safe abstraction [79]. On the basis of such an abstraction, one can compute a discrete non-blocking supervisor enforcing a language-type specification. In Sec. 4.2 (see also [78]) it has been shown that the resulting supervisor will also be non-blocking and enforce the specification when connected to the underlying continuous model (5.17), (5.18).

5.4 Monotonisation through feedback

5.4.1 Behavioural description

Let $\Sigma_{pl} = (\mathbb{N}_0, U_L, Y_L, X, \mathcal{B}_{pl}^L)$ be a dynamical system with state $X \subseteq \mathbb{R}^n$ and continuous input and output sets, $U_L \subseteq \mathbb{R}^m$, $Y_L \subseteq \mathbb{R}^n$. Moreover, assume that the system state X is trivially observable: $Y_L = X$, $\mathcal{P}_{Y_L} w = \mathcal{P}_X w$ for all $w \in \mathcal{B}_{pl}^L$.

Let us assume thatthe input space U_L can be represented as a direct product: $U_L = U_L^1 \times U_L^2$. Inputs $u_L^1 \in (U_L^1)^{\mathbb{N}_0}$ will be used for monotonisation of the plant. Furthermore, let us define a high-level input set as a finite subset of U_L^2: $U_H \subset U_L^2$, $|U_H| < \infty$. Now we can formulate the specification $\mathcal{B}_{sp}^{HL}$ using the scheme proposed in Sec. 3.2.2:

$$\mathcal{B}_{sp}^{HL} = \left\{ (u_H, y_H, u_L, y_L) \in (W_H \times W_L)^{\mathbb{N}_0} \;\middle|\; \begin{array}{l} y_H \in \mathcal{Y}_m \subset Y_H^{\mathbb{N}_0} \\ u_L \in U_L^{\mathbb{N}_0},\ \mathcal{P}_{U_L^2} u_L = u_H, \\ y_L \in Y_L^{\mathbb{N}_0}, \\ y_H(t) = y_L(t), t \in \mathbb{N}_0 \end{array} \right\}, \tag{5.19}$$

where $W_H = U_H \times Y_H$ and $W_L = U_L \times Y_L$ are the high-level and low-level signal spaces, $\mathcal{Y}_m$ is the set of high-level output signals such that for any two signals $y'_H, y''_H \in \mathcal{Y}_m$

$$\left(y'_H\big|_{t=0} \preceq y''_H\big|_{t=0}\right) \Rightarrow \left(y'_H\big|_{t=\tau} \preceq y''_H\big|_{t=\tau}\,,\ \forall \tau \in \mathbb{N}_0\right)$$

holds. If there exists a low-level controller such that the composite system plant plus controller satisfies the specification $\mathcal{B}_{sp}^{HL}$, a conservative discrete

approximation of this composite system can easily be computed. The next subsection presents a procedure for the design of a monotonising low-level controller.

5.4.2 Low-level controller design

We consider linear control systems

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) \end{cases} \tag{5.20}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^l$, B has full column rank and C has full row rank. If the monotonicity test fails, we may still be able to enforce monotonicity by appropriate feedback. For this purpose, we divide the vector of control inputs, $u' = [u_1', u_2']$, where $'$ means "transpose" and $u_1(t) \in \mathbb{R}^k, k < m$, is the part of the control input devoted to enforce monotonicity.

The system (5.20) then takes the form

$$\dot{x}(t) = Ax(x) + B^1 u_1(t) + B^2 u_2(t). \tag{5.21}$$

Defining the control input $u_1(t)$ as a linear function of the current output, $u_1(t) = Ky(t) = KCx(t)$, we change the Jacobian to $J = A + B^1 KC$ and, therefore, alter its sign structure accordingly. But we still do not have a clear algorithm to solve this problem in general because of the large number of degrees of freedom (recall that the number of orthants for an n-dimensional system is equal to 2^{n-1}).

The proposed semiformal algorithm uses an approach based on the successive reduction of the number of available degrees of freedom.

1. If either i-th row of B^1 or the j-th column of C is identical to zero, the elements $a_i j$ of the Jacobian remain unchanged.We can now check these elements for consistency by investigating whether Conditions (5.4) and (5.5) are satisfied. Clearly, if this is not the case, the monotonicity condition cannot be enforced by feedback from $y(t)$ to $u_1(t)$.

2. If the result in Step 1 is positive, we can deduce the signs of some other elements of the Jacobian from (5.7). Note the following "extreme" case: suppose, as above, that the i-th row of B^1 (resp., the j-th column of C) are zero and that all the elements in the corresponding row (resp., column) of A are nonzero (apart possibly from the entry on the diagonal). Then, the corresponding s_{ik}, $k \neq i$ (resp., s_{kj}, $k \neq j$) completely determine the required sign structure of J, as (see (5.7))

$$s_{k_1 k_2} = s_{ik_1} \oplus s_{ik_2}, \tag{5.22}$$

resp.,

$$s_{k_1 k_2} = s_{k_1 j} \oplus s_{k_2 j}. \tag{5.23}$$

If, on the other hand, elements in the corresponding row (resp., column) of A are zero, there may be several admittable orthants.

3. In the next step, we isolate the entries of the Jacobian exhibiting inappropriate signs. We now need to determine a feedback matrix K to adjust these elements without changing the signs of the other entries. For this, the elements of the real $[k \times l]$-matrix K have to satisfy

$$(-1)^{s_{qp}}(a_{qp} + \sum_{i=1}^{k}\sum_{j=1}^{l} b_{qi}^{1} k_{ij} c_{jp}) \geq 0, q \neq p.$$

The extension of the proposed algorithm to the class of nonlinear control systems is not straightforward. Usually, an arbitrary nonlinear control system admits monotonisation only in some subset of the state space, if it does at all. Let's denote by $X_\delta \subset X$ a subset of the state space X where the system can be rendered monotone w.r.t. the orthant with signature δ. It is quite common that some subspaces have nonempty intersection, i.e. $X_{\delta_1} \cap X_{\delta_2} \neq \emptyset$. Then, one must choose between several orthants. In this case, a decision can be made on the basis of heuristic considerations and can hardly be formalised. However, in some special cases (e.g. positive systems) the procedure can be successfully applied as is illustrated in the following example.

5.4.3 Example

To illustrate the applicability of the developed approach we consider a model of the biological processes in an activated sludge process, the so-called IAWQ's[2] Activated Sludge Model No.1 (see [49, 67]). This model describes the three following biological processes: removal of organic matter, nitrification, and denitrification. The considered process is an ideally mixed bioreactor with three components, which can be described by the following differential equations:

[2]International Association for Water Quality.

$$\begin{aligned}
\frac{dX_b}{dt} &= \frac{Q_{in}}{V}X_{b,in} - \frac{Q_{out}}{V}X_b + \mu(S_s)X_b - bX_b \\
\frac{dS_s}{dt} &= \frac{Q_{in}}{V}S_{s,in} - \frac{Q_{out}}{V}S_s - \frac{1}{Y}\mu(S_s)X_b \\
\frac{dS_o}{dt} &= \frac{Q_{in}}{V}S_{o,in} - \frac{Q_{out}}{V}S_o - \frac{1-Y}{Y}\mu(S_s)X_b - bX_b
\end{aligned} \tag{5.24}$$

where X_b, S_s and S_o represent the concentrations of biomass, soluble substrate and dissolved oxygen in the reactor. $X_{b,in}$, $S_{s,in}$ and $S_{o,in}$ are the influent concentrations of biomass, soluble substrate and dissolved oxygen. $\mu(S_s)$ is the specific growth rate of the biomass. It is described by Monod's equation,

$$\mu(S_s) = \frac{\bar{\mu}S_s}{K_s + S_s},$$

where $\bar{\mu}$ is the maximum specific growth rate and K_s is the half-velocity constant. The tank volume is denoted V, and the incoming and outgoing flows are Q_{in} and Q_{out}, respectively. The growth yield is Y and b is the decay rate. It is worth noting that all concentrations, input and output variables as well as parameters, are positive. Moreover, the growth yield Y is always less than one.

Using the conventional notation $u := [X_{b,in}, S_{s,in}, S_{o,in}, Q_{in}]'$ and $x := [X_b, S_s, S_o]'$ one can rewrite (5.24) as

$$\begin{aligned}
\dot{x}_1 &= \frac{u_1\, u_4}{V} - \frac{Q_{out}}{V}x_1 + \mu(x_2)x_1 - bx_1 \\
\dot{x}_2 &= \frac{u_2 u_4}{V} - \frac{Q_{out}}{V}x_2 - \frac{1}{Y}\mu(x_2)x_1 \\
\dot{x}_3 &= \frac{u_3 u_4}{V} - \frac{Q_{out}}{V}x_3 - \frac{1-Y}{Y}\mu(x_2)x_1 - bx_1.
\end{aligned} \tag{5.25}$$

The Jacobian matrix has the following form

$$\frac{Df}{Dx} = \begin{bmatrix} * & \frac{\bar{\mu}K_s x_1}{(K_s+x_2)^2} & 0 \\ -\frac{1}{Y}\frac{\bar{\mu}x_2}{K_s+x_2} & * & 0 \\ -\frac{1-Y}{Y}\frac{\bar{\mu}x_2}{K_s+x_2} - b & -\frac{1-Y}{Y}\frac{\bar{\mu}K_s x_1}{(K_s+x_2)^2} & * \end{bmatrix}. \tag{5.26}$$

We see that the partial derivatives $\frac{\partial f_1}{\partial x_2}$ and $\frac{\partial f_2}{\partial x_1}$ do not satisfy the *sign-symmetry* condition. Now one has to determine, which one has the "right" sign. The remaining elements of the Jacobian matrix satisfy conditions (5.3) and (5.4). The corresponding variables are $s_{13} = 1$, $s_{23} = 1$. Then, from (5.23) $s_{12} = 0$. This means that both $\frac{\partial f_1}{\partial x_2}$ and $\frac{\partial f_2}{\partial x_1}$ must be nonnegative. The easiest way to change the sign of $\frac{\partial f_2}{\partial x_1}$ is to use the control u_2, because

it does not enter the remaining equations. Considering the control u_2 as a function of the state variables, $u_2 = u_2(x)$, we can rewrite the Jacobian (5.26) as:

$$\frac{Df}{Dx} = \begin{bmatrix} * & \frac{\bar{\mu} K_s x_1}{(K_s + x_2)^2} & 0 \\ \frac{u_4}{V}\frac{\partial u_2(x)}{\partial x_1} - \frac{1}{Y}\frac{\bar{\mu} x_2}{K_s + x_2} & * & \frac{u_4}{V}\frac{\partial u_2(x)}{\partial x_3} \\ -\frac{1-Y}{Y}\frac{\bar{\mu} x_2}{K_s + x_2} - b & -\frac{1-Y}{Y}\frac{\bar{\mu} K_s x_1}{(K_s + x_2)^2} & * \end{bmatrix}$$

Hence, the control $u_2(x)$ has to be chosen to satisfy the following conditions:

$$\frac{\partial u_2}{\partial x_1}(x) \geq \frac{V}{u_4 Y}\frac{\mu x_2}{K_s + x_2}, \tag{5.27}$$

$$\frac{\partial u_2}{\partial x_3}(x) \leq 0, \tag{5.28}$$

$$\forall x \in \mathbb{R}^3_{\geq 0},\ u_4 \neq 0.$$

Conditions (5.27), (5.28) define a family of control laws. In particular, a control law can be chosen as

$$u_2(x) = c_1 x_1,$$

where $c_1 = \dfrac{V\bar{\mu}}{u_4^* Y}$, $u_4^* = \min u_4$. Thus, the system can be rendered monotone by a simple linear feedback.

Chapter 6

Conclusion

This work proposes a systematic approach to design and implementation of hierarchical control for complex control systems. We described a unified framework which allows for a constructive description and analysis of all levels of a hierarchical control structure. It was shown that most practically relevant classes of control systems can be described as input/output state machines. A classification of such state machines was given.

One particularly important question that arises in the hierarchical control context is to ensure that all elements of the hierarchy do not conflict. Developing the results obtained in [97], we formulated a set of constructively verifiable conditions which guarantee a non-conflicting interaction of all control levels.

To put the obtained results in a practical context, we described two particularly useful classes of intermediate control layers and analysed them with respect to the non-conflictingness property.

In the last two chapters, we considered two cases where the developed approach is applied to practically relevant control problems. In the first case, we considered an optimisation problem for a hybrid system under safety and liveness conditions. It was shown that this problem can be efficiently solved in a hierarchic way: the low-level controller enforces safety and liveness constraints while the high-level controller performs optimisation. We also showed that the action of the low-level controller can be implemented by restricting the invariants of the initial hybrid automaton. The high level uses the degree of freedom left by the low level task to find the evolution that minimises a given performance index.

In the second case, the low-level controller was designed to render the plant monotone. We discussed the question how the concept of monotonicity can be used in the context of hierarchical control. A simple and efficient algorithm to check whether an arbitrary continuous system is monotone with respect to some (a priori unknown) partial order relation was provided. This

algorithm was extended to the case of control systems. It was also shown how to enforce monotonicity with the help of feedback. The developed approach was illustrated by an example: we considered a nonlinear model of an ideally mixed bioreactor and showed that this system can be rendered monotone by a very simple linear feedback.

Bibliography

[1] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[2] D. Angeli and E.D. Sontag. Monotone control systems. *IEEE Transactions on Automatic Control*, 48(10):1684–1698, 2003.

[3] A.C. Antoulas. *Approximation of large-scale dynamical systems*. SIAM, 2005.

[4] M. Aoki. Some approximation methods for estimation and control of large scale systems. *IEEE Transactions on Automatic Control*, 23(2): 173–182, 1978.

[5] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, 2000.

[6] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43:451–476, 2007.

[7] V. Azhmyakov, S.A. Attia, D. Gromov, and J. Raisch. Necessary optimality conditionns for a class of hybrid optimal control problems. In A. Bemporad, A. Bicchi, and G. Butazzo, editors, *HSCC 2007*, LNCS 4416, pages 637–640. Springer-Verlag, 2007.

[8] A. Balluchi, L. Benvenuti, S. Engell, T. Geyer, K. H. Johansson, F. Lamnabhi-Lagarrigue, J. Lygeros, M. Morari, G. Papafotiou, A. L. Sangiovanni-Vincentelli, F. Santucci, and O. Stursberg. Hybrid control of networked embedded systems. *European Journal of Control*, (11):478–508, 2005.

[9] P. Benner, V. Mehrmann, and D.C. Sorensen, editors. *Dimension Reduction of Large-Scale Systems.*, volume 45 of *Lecture Notes in Computational Science and Engineering*. Springer, 2005.

[10] G. Birkhoff and S. Mac Lane. *Algebra*. Chelsea, 3rd edition, 1997.

[11] J.D. Boskovic, R. Prasanth, and R.K. Mehra. A multilayer control architecture for unmanned aerial vehicles. In *Proc. of the American Control Conference*, volume 3, pages 1825–1830, 2002.

[12] M.S. Branicky. *Studies in hybrid systems: modeling, analysis, and control*. PhD thesis, Massachusetts Institute of technology, 1995.

[13] M.S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical computer science*, 138 (1):67–100, 1995.

[14] R.W. Brockett. Hybrid models for motion control systems. In H.L. Trentelman and J.C. Willems, editors, *Essays in Control: Perspectives in the Theory and its Applications*, pages 29–53. Birkhäuser, 1993.

[15] G. Buttazzo, A. Bemporad, and A. Bicchi, editors. *Proceedings of the 10th International Conference: Hybrid Systems: Computation and Control, HSCC 2007*, volume 4416 of *Lecture Notes in Computer Science*, 2007. Springer.

[16] F. Caccavale, C. Natale, B. Siciliano, and L. Villani. Integration for the next generation: embedding force control into industrial robots. *IEEE Robotics & Automation Magazine*, 12(3):53–64, 2005.

[17] P.E. Caines, R. Greiner, and S.Wang. Dynamical logic observers for finite automata. In *Proc. of the 27th Conference on Decision and Control*, pages 226–233, 1988.

[18] P.S. Caines and Y.J. Wei. The hierarchical lattices of a finite machine. *Systems & Control Letters*, 25:257–263, 1995.

[19] P.S. Caines and Y.J. Wei. Hierarchical hybrid control systems: a lattice theoretic foundations. *IEEE Transactions on Automatic Control*, 43(4):501–508, 1998.

[20] C.G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[21] A. Chutinan and B. H. Krogh. Computing approximating automata for a class of hybrid systems. *Mathematical and Computer Modeling of Dynamical Systems: Special Issue on Discrete Event Models of Continuous Systems*, 6:30–50, 2000.

[22] A. Chutinan and B.H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1): 64–75, 2003.

[23] D. Corona, A. Giua, and C. Seatzu. Optimal control of hybrid automata: an application to the design of a semiactive suspension. *Control Engineering Practice*, 12:1305–1318, 2004.

[24] D. Corona, A. Giua, and C. Seatzu. Optimal feedback switching laws for autonomous hybrid automata. In *Proc. of the IEEE Intl. Symposium on Intelligent Control*, 2004.

[25] D. Corona, C. Seatzu, A. Giua, D. Gromov, E. Mayer, and J. Raisch. Optimal hybrid control for switched affine systems under safety and liveness constraints. In *IEEE International Symposium on Computer Aided Control Systems Design*, pages 35–40, 2004.

[26] D. Corona, A. Giua, and C. Seatzu. Stabilization of switched system via optimal control. In *Proc. of the 16th IFAC World Congress*, 2005.

[27] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control*, 43(4):564–568, 1998.

[28] A. Dasgupta, B. Pandurangan, R.G. Landers, and S.N. Balakrishnan. Hierarchical optimal control of a turning process - linearization approach. In *Proc. of the American Control Conference*, volume 3, pages 2608–2613, 2003.

[29] J.M. Davoren, T. Moor, and A. Nerode. Hybrid control loops, A/D maps, and dynamic specifications. In C.J. Tomlin and M.R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2002.

[30] A. D'Innocenzo, A.A. Julius, M.D. Di Benedetto, and G.J. Pappas. Approximate timed abstractions of hybrid automata. In *Proc. of the 46th IEEE Conf. on Decision and Control*, pages 4045–4050, 2007.

[31] S. Diop. Elimination in control theory. *Mathematics of Control, Signals, and Systems*, 4(1):17–32, 1991.

[32] D.P. Eckman and I. Lefkowitz. Principle of model techniques in optimizing control. In J.F. Coales, J.R. Ragazzini, and A.T. Fuller, editors, *Proc. of the First IFAC World Congress*, volume II, pages 970–976, 1960.

[33] M. Egerstedt and B. Mishra, editors. *Proceedings of the 11th International Conference: Hybrid Systems: Computation and Control, HSCC 2008*, volume 4981 of *Lecture Notes in Computer Science*, 2008. Springer.

[34] W. Findeisen, F.N. Bailey, M. Brdyś, K. Malinowski, P. Tatjewski, and A. Woźniak. *Control and coordination in hierarchical systems*, volume 9 of *International Series on Applied Systems Analysis*. John Wiley & Sons, 1980.

[35] R.G. Franks and C.W. Worley. Quantitative analysis of cascade control. *Ind. Eng. Chem.*, 48(6):1074–1079, 1956.

[36] M. Garavello and B. Piccoli. Hybrid necessary principle. In *Proc. of the 44th IEEE Conf. on Decision and Control*, pages 723–728, 2005.

[37] S. Geist, D. Gromov, and J. Raisch. Timed discrete event control of parallel production lines with continuous outputs. *Discrete Event Dynamic Systems*, 18(2):241–262, 2008.

[38] A. Girard and G.J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45:566–571, 2009.

[39] D.N. Godbole, J. Lygeros, and S. Sastry. Hierarchical hybrid control: a case study. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 166–190. Springer, 1995.

[40] R. Goebel, R.G. Sanfelice, and A.R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[41] K.A. Grasse. Admissibility of trajectories for control systems related by smooth mappings. *Mathematics of Control, Signals, and Systems*, 16:120–140, 2003.

[42] D. Gromov and J. Raisch. Detecting and enforcing monotonicity for hybrid control systems synthesis. In *Proc. of the 2nd IFAC Conference on Analysis and Design of Hybrid systems, ADHS'06*, pages 395–402, 2006.

[43] D. Gromov, E. Mayer, J. Raisch, D. Corona, C. Seatzu, and A. Giua. Optimal control of discrete-time hybrid automata under safety and liveness constraints. In *Proc. IEEE International Symposium on Intelligent Control. Mediterranean Conference on Control and Automation*, pages 243–249, 2005.

[44] D. Gromov, S. Geist, and J. Raisch. Timed discrete event control of a parallel production line with continuous output. In *Proc. of the 2nd IFAC Conference on Analysis and Design of Hybrid systems, ADHS'06*, pages 205–211, 2006.

[45] R.L. Grossman and R.G. Larson. An algebraic approach to hybrid systems. *Theoretical computer science*, 138:101–112, 1995.

[46] H Guéguen and J. Zaytoon. On the formal verification of hybrid systems. *Control Engineering Practice*, 12:1253–1267, 2004.

[47] W. Hahn. *Stability of motion*. Springer, 1967.

[48] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Transactions on Automatic Control*, 47(9):1536–1540, 2002.

[49] M. Henze, C.P.L. Grady, W. Gujer, G.v.R Marais, and T. Matsuo. *Activated sludge model No.1*. IAWPRC Scientific and technical reports, No.1. International Association on Water Pollution Research and Control, London, 1987.

[50] A. Itigin, J. Raisch, T. Moor, and A. Kienle. A two-level hybrid control strategy for the start-up of a coupled distillation plant. In *Proc. of the European Control Conference (ECC2003)*, 2003.

[51] K. H. Johansson. Hybrid control systems. In H. Unbehauen, editor, *Encyclopedia of Life Support Systems (EOLSS), Theme 6.43: Control Systems, Robotics and Automation, Developed under the Auspices of the UNESCO*. Eolss Publishers, 2004. URL `http://www.eolss.net`.

[52] K. H. Johansson, J. Lygeros, and S. Sastry. Modeling of hybrid systems. In H. Unbehauen, editor, *Encyclopedia of Life Support Systems (EOLSS), Theme 6.43: Control Systems, Robotics and Automation, Developed under the Auspices of the UNESCO*. Eolss Publishers, 2004. URL `http://www.eolss.net`.

[53] A.A. Julius and A.J. van der Schaft. State maps of general behaviors, their lattice structure and bisimulations. In *Proceedings of the Sixteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2004)*, 2004.

[54] A.A. Julius, S.N. Strubbe, and A.J. van der Schaft. Control of hybrid behavioural automata by interconnection. In *Proc. IFAC Conference on the Analysis and Design of Hybrid Systems (ADHS'03)*, pages 135–140, 2003.

[55] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002.

[56] B. Khoussainov and A. Nerode. *Automata theory and its applications*. Birkhäuser, 2001.

[57] M. Kokar and K. Baclawski. Modeling combined time- and event-driven dynamic systems. In *Proc. 9th OOPSLA Workshop on Behavioral Semantics*, pages 112–129, 2000.

[58] X.D. Koutsoukos and P.J. Antsaklis. Safety and reachability of piecewise linear hybrid dynamical systems based on discrete abstractions. *Discrete Event Dynamic Systems*, 13(3):203 – 243, 2003.

[59] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7): 1026–1049, 2000.

[60] C.P. Kwong and C.F. Chen. A quotient space analysis of aggregated models. *IEEE Transactions on Automatic Control*, 27(1):203–205, 1982.

[61] C.P. Kwong and Y.-K. Zheng. Aggregation on manifolds. *Int. J. Systems Sci.*, 17(4):581–589, 1986.

[62] T. Larsson and S. Skogestad. Plantwide control - A review and a new design procedure. *Modelling, Identification and Control*, 21(4): 209–240, 2000.

[63] J.M. Lee. *Introduction to smooth manifolds*. Springer, 2003.

[64] E.S. Lemch and P.E. Caines. Hierarchical hybrid systems: Partition deformations and applications to the acrobot system. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 1998.

[65] E.S. Lemch and P.E. Caines. Hybrid partition machines with disturbances: hierarchical control via partition machines. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 5, pages 4909 – 4914, 1999.

[66] C.M. Libosvar. Hierarchies in production management and control: A survey. Technical Report LIDS-P-1734, Laboratory for Information and Decision Systems, MIT, 1988.

[67] C.-F. Lindberg. *Control and estimation strategies applied to the activated sludge process*. PhD thesis, Uppsala University, 1997.

[68] J. Lunze and F. Lamnabhi-Lagarrigue, editors. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, 2009.

[69] J. Lunze and B. Nixdorf. Representation of hybrid systems by means of stochastic automata. *Mathematical and Computer Modeling of Dynamical Systems*, 4:383–422, 2001.

[70] J. Lygeros, K.H. Johansson, S.N. Simić, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

[71] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture notes in computer science*, pages 403–417. Springer, 2001.

[72] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and computation*, 185(1):105–157, 2003.

[73] M.S. Mahmoud. Multilevel systems control and applications: a survey. *IEEE Transactions on System, Man, and Cybernetics*, 7(3):125–143, 1978.

[74] R. Majumdar and P. Tabuada, editors. *Proceedings of the 12th International Conference: Hybrid Systems: Computation and Control, HSCC 2009*, volume 5469 of *Lecture Notes in Computer Science*, 2009. Springer.

[75] I. Mareels and J.C. Willems. Elimination of latent variables in real differential algebraic systems. In V.D. Blondel, E.D. Sontag, M. Vidyasagar, and J.C. Willems, editors, *Open Problems in Mathematical Systems and Control Theory*, pages 141–147. Springer, 1998.

[76] P. Melloge and P. Kachroo. *Model Abstraction in Dynamical Systems: Application to Mobile Robot Control*, volume 379 of *Lecture Notes in Control and Information Sciences*, chapter 5. Abstraction, pages 61–80. Springer, 2008.

[77] M.D. Mesarović, D. Macko, and Y. Takahara. *Theory of hierarchical, multilevel, systems.* Academic Press, 1970.

[78] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Systems & Control Letters*, 38:157–166, 1999.

[79] T. Moor and J. Raisch. Abstraction based supervisory controller synthesis for high order monotone continuous systems. In S. Engell, G. Frehse, and E. Schneider, editors, *Modelling, Analysis, and Design of Hybrid Systems*, volume 279 of *Lecture notes in control and information sciences*, pages 247–265. Springer, 2002.

[80] T. Moor, J. Raisch, and J.M. Davoren. Admissibility criteria for a hierarchical design of hybrid control systems. Technical report, Departament of Systems Engineering, RSISE, Australian National University, 2002. A full version of [82].

[81] T. Moor, J. Raisch, and S. O'Young. Discrete supervisory control of hybrid systems based on ℓ-complete approximations. *Discrete Event Dynamical Systems: Theory and Applications*, 12:83–107, 2002.

[82] T. Moor, J. Raisch, and J.M. Davoren. Admissibility criteria for a hierarchical design of hybrid control systems. In *Proc. IFAC Conference on the Analysis and Design of Hybrid Systems (ADHS'03)*, pages 398–394, 2003.

[83] T. Moor, J. Davoren, and J. Raisch. Learning by doing - systematic abstraction refinement for hybrid control synthesis. *IEE Proc. Control Theory & Applications*, 153:591–599, 2006. special issue on hybrid systems.

[84] A.S. Morse. Control using logic-based switching. In A. Isidori, editor, *Trends in Control: A European Perspective*, pages 69–113. Springer, 1995.

[85] J. Munkres. *Topology*. Prentice Hall, 1999.

[86] H. Nijmeijer and A.J. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer, 1990.

[87] G. Obinata and B.D.O. Anderson. *Model reduction for control system design*. Springer, 2001.

[88] C.M. Özveren and A.S. Willsky. Observability of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.

[89] F. De Paoli and F. Tisato. On the complementarity nature of event-driven and time-driven models. *Control Engineering Practice*, 4(6): 847–854, 1996.

[90] G.J. Pappas and S. Sastry. Towards continuous abstractions of dynamical and control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 329–341. Springer, 1997.

[91] G.J. Pappas and S. Simić. Consistent abstractions of affine control systems. *IEEE Transactions on Automatic Control*, 47(5):745–756, 2002.

[92] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6): 1144–1159, 2000.

[93] J. Park, J. Obeysekera, and R. VanZee. Multilayer control hierarchy for water management decisions in integrated hydrologic simulation model. *Journal of Water Resources Planning and Management*, 133 (2):117–125, 2007.

[94] J.L. Piovesan, H.G. Tanner, and C.T. Abdallah. Discrete asymptotic abstractions of hybrid systems. In *Proc. of the 45th IEEE Conf. on Decision and Control*, pages 917–922, 2006.

[95] J.W. Polderman and J.C. Willems. *Introduction to Mathematical Theory. A behavioural approach*, volume 26 of *Texts in Applied Mathematics*. Springer, 1998.

[96] J. Raisch. Discrete abstractions of continuous systems – an input/output point of view. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):6–29, 2000. Special issue on Discrete Event Models of continuous systems.

[97] J. Raisch and T. Moor. Hierarchical hybrid control synthesis and its application to a multiproduct batch plant. In T. Meurer, K. Graichen, and E.D. Gilles, editors, *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*, volume 322 of *Lecture notes in control and information sciences*, pages 199–216. Springer, 2005.

[98] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):569–573, 1998.

[99] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

[100] M.S. Ravi and J.Rosenthal. A general realization theory for higher-order linear differential equations. *Systems & Control Letters*, 25: 351–360, 1995.

[101] J.H. Sandee. *Event-driven control in theory and practice. Trade-offs in software and control performance*. PhD thesis, Technische Universiteit Eindhoven, 2006.

[102] J.H. Sandee, W.P.M.H. Heemels, and P.P.J. van den Bosch. Case studies in event-driven control. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 762–765. Springer, 2007.

[103] J. Schröder. *Modelling, State observation and Diagnosis of Quantised Systems*. Springer-Verlag, 2002.

[104] B. De Schutter and W.P.M.H. Heemels. Modelling and control of hybrid systems. Lecture notes of the DISC Course, September 2004.

[105] C. Seatzu, D. Corona, A. Giua, and A. Bemporad. Optimal control of continuous-time switched affine systems. *IEEE Transactions on Automatic Control*, 51(5):726–741, 2006.

[106] C. Seatzu, D. Gromov, J. Raisch, D. Corona, and A. Giua. Optimal control of discrete-time hybrid automata under safety and liveness constraints. *Nonlinear analysis*, 65:1188–1210, 2006. Special issue on hybrid systems and applications (5).

[107] M.S. Shaikh and P.E. Caines. On the hybrid optimal control problem: Theory and algorithms. *IEEE Transactions on Automatic Control*, 52 (9):1587–1603, 2007.

[108] M.G. Singh. *Dynamical hierarchical control*. North-Holland, 2nd edition, 1980.

[109] M.G. Singh and K. Hindi. A multilevel multilayer framework for manufacturing control. *Journal of intelligent and robotic systems*, 4:75–93, 1991.

[110] H.L. Smith. *Monotone dynamical systems: an introduction to the theory of competitive and cooperative systems*, volume 41 of *Mathematical surveys and monographs*. American Mathematical Society, Providence, RI, 1995.

[111] J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. Interface and controller design for hybrid control systems. In P.J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 462–492. Springer, 1995.

[112] T. Stoilov and K. Stoilova. *Noniterative Coordination in Multilevel Systems*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, 1999.

[113] P. Tabuada. Symbolic models for control systems. *Acta Informatica*, 43:477–500, 2007.

[114] P. Tabuada, G.J. Pappas, and P. Lima. Compositional abstractions of hybrid control systems. *Discrete Event Dynamic Systems*, 14(2): 203–238, 2004.

[115] P. Tatjewski. Advanced control and on-line process optimization in multilayer structures. *Annual Reviews in Control*, 32:71–85, 2008.

[116] M.A Temchin and D.J. Bell. A modified elimination procedure in nonlinear algebraic control theory. *IMA Journal of Mathematical Control and Information*, 10(3):195–204, 1993.

[117] A.J. van der Schaft. On realization of nonlinear systems described by higher-order differential equations. *Mathematical systems theory*, 19: 239–275, 1987.

[118] A.J. van der Schaft. Equivalence of dynamical systems by bysimulation. *IEEE Transactions on Automatic Control*, 49(12):2160–2172, 2004.

[119] Y. Wang and E.D. Sontag. Generating series and nonlinear systems: analytic aspects, local realizability, and I/O representations. *Forum Mathematicum*, 4:299–322, 1992.

[120] J.C. Willems. Frome time series to linear system - Part I. Finite dimensional linear time invariant systems. *Automatica*, 22(5):561–580, 1986.

[121] J.C. Willems. Models for dynamics. *Dynamics Reported*, 2:172–269, 1989.

[122] J.C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36:258–294, 1991.

[123] J.C. Willems. Open dynamical systems and their control. In *Documenta Mathematica*, pages 697–706. DMV, 1998. Extra Volume ICM III.

[124] W.M. Wonham. Supervisory control of discrete-event systems. On-line monograph, 2008.

[125] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134, 1990.