TECHNISCHE
UNIVERSITÄT
WIEN

D I S S E R T A T I O N

Mathematical Characterisation of State Events in Hybrid Modelling

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Doktors der technischen Wissenschaften

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

E101

Institut für Analysis und Scientific Computing

eingereicht an der Technischen Universität Wien

Fakultät für Mathematik und Geoinformation

von

Dipl.-Ing. Dipl.-Ing. Andreas Körner, BSc.

0225089

Donaufelderstraße 57/2/8,
1210 Wien.

Wien, am 13. Mai 2015 . . . .

Kurzfassung

Die Doktorarbeit befasst sich mit dem Thema Mathematische Charakterisierung von Zustandsereignissen in hybrider Modellbildung. Der Aufbau der Arbeit ist gegliedert in drei wesentliche Aspekte.

Der erste Teil der Arbeit widmet sich der Definition von dynamischen hybriden Modellen bzw. dynamischen hybriden Systemen. Dabei wird eine klassische Systemtheorie für DAE Modelle definiert, welche die kontinuierliche Dynamik eines Zustandes beschreibt. Diese wird mit einer diskreten Dynamik des gesamten hybriden Modells, in Form eines diskreten Automaten, vereint und damit die erste Definition eines dynamischen hybriden Modells gegeben.

An die erste Definition schließt ein Kapitel an, welches typische hybride Phänomene betrachtet und die gegebene Definition hinsichtlich dieser Phänomene prüft. Im Einzelnen wird überprüft, welche Phänomene in der gegebenen Definition erfasst werden können und für welche eine Verallgemeinerung der mathematischen Umgebung notwendig ist. Diese Verallgemeinerungen werden anschließend dann auch vorgenommen.

Nachdem die mathematische Basis definiert und überprüft wurde, werden die numerischen Aspekte betrachtet, welche von einer Simulationsumgebung bewältigt werden müssen. Die definierte mathematische Umgebung wird in Relation zu den Aufgaben des Simulators gesetzt. Der numerische Ablauf von Zustandsereignissen und deren Folgen werden betrachtet. Die notwendige Struktur eines Simulators wird beleuchtet und numerische Algorithmen zur Erkennung und Auffindung von Zustandsereignissen werden vorgestellt.

Der zweite Teil der Arbeit ist, aufbauend auf der definierten und angepassten mathematischen Struktur, die Charakterisierung der Zustandsereignisse. Diese werden eingeteilt in die übergeordneten Kategorien einer Änderung eines Wertes oder Werte eines Vektors, strukturelle Änderungen in der Beschreibung oder einer vollkommenen Modelländerung der kontinuierlichen Dynamik. Einige Beispiele werden diese Charakterisierung untermauern, die mathematische Definition der Charakterisierung ist angepasst an den Formalismus der Systemtheorie.

Die Charakterisierung ist eine rein mathematische Aufgabe. Die Beispiele zur Veranschaulichung, sowie zur Formulierung gewisser Anforderungen sind meist technischer Natur und kommen aus Gebieten der Naturwissenschaften. Eines dieser Beispiele ist der Schwingkreis mit Diode, welcher ein gutes Beispiel für die Untersuchung möglicher Zustandsänderungen in technischen Systemen darstellt.

Das grundsätzliche Problem in der Modellbildung ist die Tatsache, dass es keine eindeutigen Modelle gibt. Die mathematische Charakterisierung erfüllt aber zumindest die Anforderung, dass die Zustandsänderungen nicht mehr auf Basis einer Implementierung diskutiert werden müssen, sondern eine mathematische Grundlage auf Basis eines Modells erlaubt.

Der letzte Teil der Arbeit ist ein weiterer Schritt der Verallgemeinerung. Im Bereich der mathematischen Modellbildung und Simulation gibt es eine Vielzahl an Methoden, die es erlauben eine kontinuierliche Dynamik zu beschreiben. Der letzte Schritt der Arbeit nimmt eine Verallgemeinerung in diese Richtung vor. Es werden nicht mehr ausschließlich DAE Modelle verwendet um die Dynamik in einem Zustand zu beschreiben, auch andere Methoden sind erlaubt. Dieser Umstand wird als Multi-Methoden Modell bezeichnet, was aber nicht auf die Koexistenz von Submodellen unterschiedlicher Methodik bezogen ist, sondern auf die verschiedenen vorhandenen Methoden innerhalb eines dynamischen hybriden Modells. Methoden wie zelluläre Automaten, agentenbasierte Ansätze oder andere Dynamiken können so vereint in einem hybriden Modell arbeiten. Lediglich die Anforderungen der diskreten Dynamik, die nach wie vor die gleiche ist, müssen erfüllt werden.

Ein Ausblick am Ende der Arbeit skizziert die weiteren Möglichkeiten eine ähnliche Charakterisierung für andere Beschreibungen einer kontinuierlichen Dynamik zu finden sowie auch stochastische Aspekte in die Betrachtungen zu integrieren.

Abstract

This is a thesis about the mathematical characterisation of state events in the context of dynamical hybrid models. The structure of the thesis is designed in three major aspects.

The first part addresses the mathematical definition of dynamical hybrid models, respectively dynamical hybrid systems. For this purpose, the system theory of a dynamical system described by a differential algebraic equation (DAE) is introduced. The following step combines this theory to a classical discrete dynamics, associated with a discrete automaton. In this thesis the hybrid dynamical model or system is defined as automaton, where the states are characterised via a corresponding DAE system.

After the definition of the mathematical background, classical hybrid phenomena will be addressed and the given mathematical environment, regarding these phenomena, is reviewed. Several items are covered and for some other, a generalisation of the structure is necessary, which is done in the end of this chapter.

Following, several numerical aspects in the simulation environment, concerning state events, will be addressed. Aspects like the detection and location of events and the corresponding event actions in the simulation environment are discussed to relate the mathematical issues to the duties of a certain simulator. Furthermore, classical location algorithms and special cases of the mathematical description are presented.

The second part of the thesis focuses on the characterisation of the state events after being defined in the first part. The characterisation will be motivated on particular examples and it will be defined in an abstract mathematical environment of a hybrid dynamical system as well as in the system simulation formulation. Different aspects will be discussed: the simple change of values like a parameter, the change of the continuous dynamics as in a switching system, the change of the dimension of the state space up to the complete model change. The classification of events will be done with respect to their complexity of change.

This classification part will give some introducing motivations of the different state events illustrated by some examples. The mathematical definition will be given in abstract environment formalism, matching to the formalism of system simulation.

The characterisation of state events is a mathematical task. Nevertheless, the thesis will be accompanied by several examples in the field of engineering and natural sciences. Several are academic examples which means, the solution can be given in an analytical meaning to verify and review the mathematical modelling outcomes. One prominent example is a modified oscillating circuit from electrical engineering. This oscillating circuit is combined with a diode which gives the switching part in the circuit. Depending on the based mathematical model of the diode, the example shows various state events.

One problem of mathematical modelling in principal is the uniqueness of a model. This uniqueness is not given and as a result, also modelling of state events and their characterisation is not unique. The corresponding model implies several realisations of the state event or better to say in the realisation of the corresponding definition and formulation of the event.

The mathematical characterisation of the state events provides a mathematical foundation to discuss state events on basis of a model without concerning the implementation of state events.

The end of the work will be a last generalisation, an outlook for further work. The given characterisation has been done in the context of dynamic hybrid systems with DAE models for the continuous dynamics. These systems are discrete systems, where the nodes are described by DAEs. The last extension focuses on the integration of different modelling techniques. The description of the continuous dynamics will be assumed to be more generalised to allow more than DAE specifications. In this context, each state could contain a model which is not the same model structure like in the other node. In one node there can be a cellular automaton, in the other, a DAE or an Agent Based model and in a further one, another structure of a dynamical system can be used. The only condition is that the models in the nodes respect the requirement of the mathematical description in terms of state vector, input and output variables, and so on. The characterisation of events in this setup should be the same, only the calculation of the state variables would be done not always by a DAE but also from different model structures.

Further work may focus on this work environment and create a similar mathematical description for multi method models, as it is introduced in this thesis about dynamical hybrid systems. Also stochastic approaches are not covered in this thesis, which can be interesting for further work.

Acknowledgment

Writing a doctoral thesis is nothing self-evident. For this reason, I would like to thank my professor and supervisor Felix Breitenecker for offering me this possibility. He taught me during this time much more than mathematical modelling and simulation, especially in personal issues I have learned a lot.

The process of writing the thesis is demanding and requires a lot. I sincerely would like to express my gratitude to my dearest Genta. She passed with me the last part of this time, accepted my mood, stress and helped me to bring my English in the right form.

Moreover, thanks to Stefanie. She took care on parts of my responsibilities at work and because of her I had more time to finish my thesis. Thank you as well for proofreading my work.

Last, but not least I would like to thank my family, especially my parents, who enabled my academic development and supported me from the beginning of my study.

Vienna, May 2015 Andreas Körner

If I were again beginning my studies,
I would follow the advice of Plato
and start with mathematics.

Galileo Galilei

Contents

Chapter 1

Introduction

Mathematical characterisation of state events in hybrid modelling requires a detailed localisation of this three addressed aspects. The term hybrid is in the field of modelling and simulation, as well as in control theory, a frequently used key word. In the following section of this introduction chapter, hybrid models will be discussed and and the scope of the thesis will be addressed. Also the starting point and the aim of the thesis shall be defined.

As a following step, a section will introduce related issues. Not only traditional literature will be cited, also simulation software which are related to hybrid models, will be discussed. Due to the fact that hybrid models are used as well in other disciplines, a short excursus in related topics will be given.

The last section of the introduction will give an outline of the thesis and an explanation of the structure, the thesis is following.

1.1 Hybrid Models and State Event Modelling

Hybrid Models are a special case in mathematical modelling theory. The term hybrid is used in many different ways, partly to describe a model structure and partly the understanding is mixed with the simulation or the implementation in a particular environment. This section will separate different understandings and make clear which aspects are covered in this thesis.

1.1.1 Terminology

A hybrid model structure starts from a certain behaviour under a particular condition and changes this behaviour if the condition is no longer fulfilled. Several disciplines offer behaviour of particular environments or systems, which fulfill this linguistic definition. The change from one area of validity is a change of the model and the changeover to the other behaviour, under the particular condition, is the event in a state space model.

Hybrid Models. This thesis follows the classification of hybrid models in the context of dynamical systems distinguished by discretisation of value and time. The value related to dynamical systems is called the state, which is assumed to be a functional relation depending on the time.

Note 1.1. Because of the mixture of nomenclature, on the one hand hybrid models and dynamical systems and on the other hand, the difference between a model and a system is not strictly executed in literature. Also in this thesis, the term model and system will be used as equal. If the distinction is necessary, it will be emphasised in the particular situation.

Figure 1.1 illustrates the separation between continuous and discrete state characteristics over time. In this listing, hybrid model behaviour is shown.

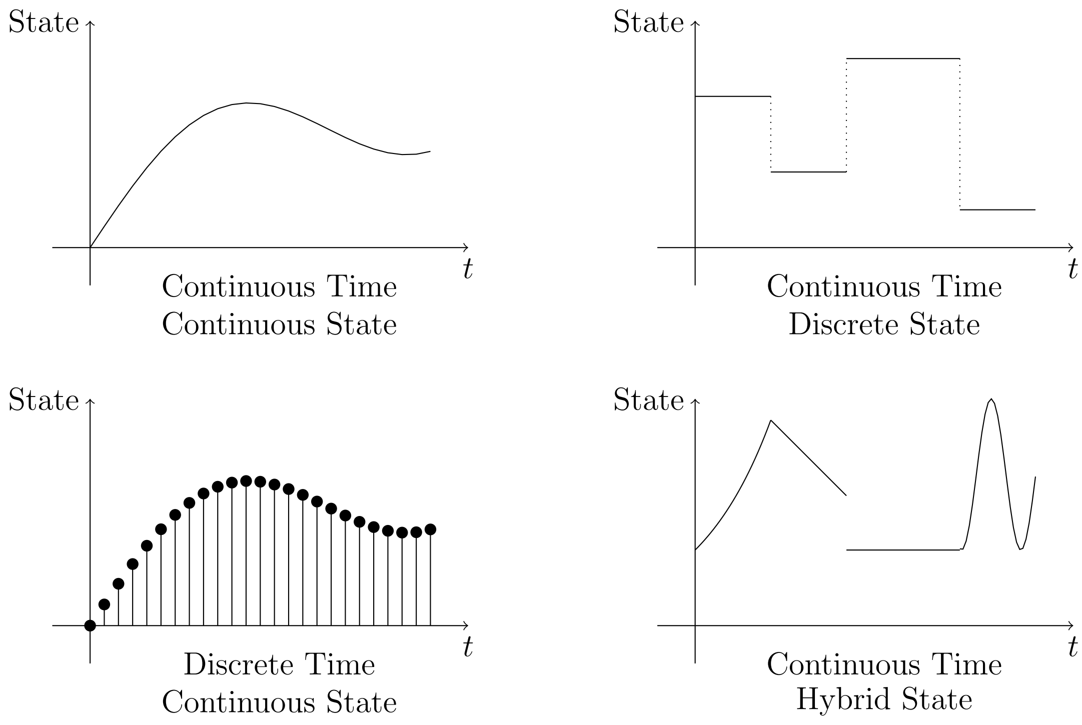


Figure 1.1: Overview about different Time Characteristics for Classification of Dynamic Systems.

State Event Modelling. This modelling approach provides the corresponding mathematical description in that way, that the (numerical) computation starts with a certain dynamical system and when the moment of change occurs, a state event changes this mathematical description. In the setting, the state is described by a plain dynamical system of the form

$$\dot{x} = f(x, t).$$

State event modelling changes this description, in the moment the change occurs to another dynamical system description.

To summarise, the both terms, hybrid modelling and state event modelling, address the same range of model description. The point of view is a bit different but these modelling methods describe the same process; to model sequential processing of model description. State event modelling, normally restricts the range of model description to ordinary differential equations or algebraic differential equations, whereas the term hybrid modelling also occurs in different model description approaches and methodology.

As a final remark, in the subsection reserved for terminology, also a delimitation should be mentioned. The terms hybrid and state event models in this thesis are not used to express the use of submodels with different kind of model description to form a overall model and neither multi–method models, where a model of a rather complex structure or system is split in different submodels, where several problems are solved with different types of modelling approaches, as presented in [3]. This modelling approach is often referred as multi–method simulation approach, because the models which are considered in this submodels are motivated by the implementation w.r.t. complexity, difficulties in implementation, simulation time, etc. The present thesis deals with the mathematical environment of sequential processed DAE subcomponents of a hybrid model which will be extended in a last step to multi–method subcomponents. Nevertheless, still the sequential processing defines this structural model, no parallel working submodel are addressed with this approach.

1.1.2 Starting Point and Aim of the Thesis

Based on the facts, which kind of hybrid models are considered, as presented in the subsection before, another view is influencing the work. The surroundings which will be considered for the modelling approach is the field of *system simulation*. The aim of system simulation is the procedural implementation and execution of a simulation run in a certain environment with appropriate models. Figure 1.2 illustrates the procedure as a consecutively processing of model descriptions over time.

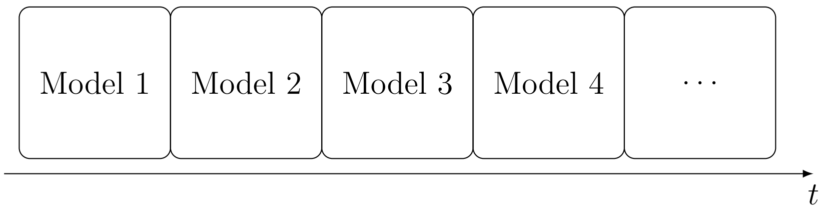


Figure 1.2: Hybrid Model in the Context of System Simulation.

This surrounding is simultaneously the problem. Usually in the modelling process the simulation environment is respected. This means, that most of the models are partly described in a mathematical formalism, but especially the changes to other descriptions are oriented to the implementation and partly only specified by an algorithmic definition. This eliminates the benefit of a mathematical abstract model definition and the possibilities to compare different model strategies, degree of complexity, etc. regarding the simulation tasks. Also a correct model analysis, as aspects of stability or concerns of control theory, are hard to perform.

As an illustration of the problem definition, an example in the common simulation environment MATLAB will be discussed. MATLAB offers a possibility to change the description of dynamical systems while executing the numerical solver, see [48]. MATLAB provides a function handle that includes one or more event functions of the form

```
[value,isterminal,direction] = events(t,y).
```

This option provides the possibility of stopping a numerical calculation, if one or more so-called events take place. There are several attributes offered, of which events are respected, the direction of crossing the zero can be parametrised and the value of the event function is available. In [48] several examples are presented, where this function is used to change the descriptive ordinary differential equation.

Exactly this introducing example represents the problem. The definition of an event is done on the level of implementing. Either in a command line oriented simulation environment by programming, or in a graphical oriented simulation environment by structuring the simulation model. In the field of system simulation, the state event modelling approach associated with a mathematical layer is missing.

The Aim of the thesis is to define and analyse a structured model environment, which respects the procedural aspect of system simulation, as illustrated in Figure 1.2, and offers a mathematical environment. This environment can be assigned to the numerical duties but the model structure is not influenced by limitations or bounds of a particular simulation environment. The thesis will not discusses the modelling process up to the stage of a state event model, respectively to a (procedural) hybrid model. The thesis will start with this precondition.

1.2 Related Aspects

This section cites detailed publications, books and theses which are related to the topic of this thesis. Several books and articles are mentioned which contribute in the final presentation of this thesis. Due to its scope, as well software is reviewed. The field of mathematical modelling is closely related to simulation environment which finally realises the implemented simulation run of this models. This aspect guides to collecting some ideas and review some simulation environments and software.

1.2.1 Related Work

Several books are helpful to get access in the field of mathematical modelling and hybrid models. For foundations in mathematical modelling theory and dynamic systems, the books [28] and [29] give introduction to mathematical system theory as well as differential equations related to dynamical systems.

One of the standard work, in the field of mathematical modelling and simulation, is the book continuous system simulation, [10]. This book is offering a wide overview about applied modelling with theoretic background. It is covering several solver types for simulation environment and introduces several modelling and simulation methods and techniques.

Another book, which offers a wide range of modelling techniques and works as a handbook for modelling and simulation issues is [18].

For a first access in the subject of hybrid systems especially the book [50], [21] and [15] are useful. [50] deliveres the foundation for the definition of the mathematical environment in chapter 2 and [40] deliveres ideas in structuring the present thesis.

Beside the books and articles, one thesis is useful to enter in several questions concerning the subject. The thesis [20] is out of an engineering field and gave the first inspiration.

Several publications were consulted in the thesis writing process, some have the character to understand a range of sub classified model description. Especially the aspects of switched systems, as discussed in [53] and [31], helped in checking the mathematical environment regarding hybrid phenomena, as addressed in chapter 3. Also the chapter in a book of various applications, as [37] was helpful in this regard. Concluding this publications and needs for a certain simulation environment, the structure for a generalised mathematical framework for using the characterisation of state event modelling was introduced in [4].

Important research in the field of state events is done in numerical issues. Especially the detection and location of state events are fields several publications are dealing with. A good overview and introduction in several strategies from the numerical environment point of view is given in [16].

Starting from the mentioned work, the present thesis developed in the present structure. Further used bibliography is mentioned in the corresponding chapter or section.

1.2.2 Related Software

Hybrid modelling is always related to the question of proper simulation environments, which can deal with hybrid models. As discussed in the beginning of this chapter, MATLAB is offering a certain event environment for ODE solver, on an algorithmic level. MATLAB is a certain standard, therefore the first comment is dedicated to this software tool.

Beside this standard MATLAB, there are several other toolboxes in MATLAB and software which are oriented to hybrid modelling. In this subsection, a small variety will be presented, which is mentioned in several publications related to the topic of the thesis.

To provide a structured access to this issues, the first software is HYSDEL. *HYSDEL – Hybrid System DEscription Language* allows to form simulation models for a specialised case, described by interconnections of linear dynamic systems, automata, if–then–else and propositional logic rules. An overview about the simulation language is offered by the manual [49].

From the same group at ETH Zürich which was involved in HYSDEL the Multi-Parametric Toolbox is provided. This toolbox is a Matlab-based toolbox for parametric optimisation, computational geometry and model predictive control. On basis of this toolbox the *HIT – Hybrid Identification Toolbox* is available, see [17], which deals with regression with Piece-Wise Affine (PWA) maps and identification of Piece-Wise AutoRegressive Exogenous (PWARX) models. More related is another MATLAB Toolbox, the *Hybrid Toolbox for MATLAB*. The toolbox deals with modelling, simulating, and verifying hybrid dynamical systems. HYSDEL models are available in MATLAB and mixed logical dynamical (MDL) systems are handled, see [1]. The MDL objects can be converted in piecewise affine (PWA) objects, which are piecewise affine discrete–time dynamical system in the form

$$x_{k+1} = A_i x_k + B_i u_k + f_i.$$

This is the discrete–time state space description with an additive term f_i, which characterises the model as affine. The index i indicates different descriptions in between the system switches.

Related to this class of hybrid models and software, several publications deal with this issue in detail. [43] deals with the translation of hybrid automata into linear piecewise affine models and [45] addresses modelling of hybrid systems by mixed logical dynamical systems.

1.2.3 Related Topics

Beside related work and related software, the current subsection will give a rough overview about related topics which are partly related to the topic of the thesis. Mainly, the related topics are located in control engineering and control theory, identification, optimisation, analysis of stability of hybrid systems and predictive methods.

Control topics with an alternative approach via neuronal networks are discussed in [42]. Issues discussed regarding model predictive control with discrete inputs are covered in [44], hybrid state space approaches in [22].

As well some stochastic related topics are known, which offer interesting ideas, as in [9], but stochastic aspects are in this thesis not covered.

1.3 Structure of the Thesis

Before entering in the different subjects of the following chapter and sections, the aim of this final section of the introduction is to give an idea how the structure of the thesis is designed.

Introduction. The role of the introduction is clear. Nomenclature, related work and topics as well as a structure of expectation is presented.

Mathematical Framework for Dynamic Hybrid Models. In this chapter the mathematical start is defined. The first version, in a sequence of redefinition and refinements in definitions, shall provide a proper mathematical foundation to deal with hybrid models on the one hand and state event models on the other hand. Hybrid models will be defined by merging a automaton structure with several DAE model structures. State event modelling is, as introduced before, closely related to a simulation environment. In this chapter, the basics of state event modelling will be given, without including numerical aspects of a simulation environment. This aspect will be added later in the corresponding chapter.

Classification and Generalisation of Dynamic Hybrid Systems. After definition of the foundations, several hybrid phenomena will be discussed and the given structure will be analysed regarding the capability to cover this aspects. If some aspects are not covered, the mathematical framework will be enlarged and refined up to a point where all aspects, relevant for the thesis, are provided.

State Event Handling in the Numerical Environment. This chapter continues with the state event handling procedure of chapter 2, including here also the aspects which have to be covered from a simulation point of view. Tasks and duties of a simulation environment will be discussed and some considerations w.r.t. numerical issues have to be addressed. This chapter is dedicated to the interface between the mathematical and the simulation environment.

Mathematical Characterisation of State Events. After several chapters of defining the foundations for the mathematical requirements and link to the simulation aspects, this chapter introduces and formalises the state event actions or transitions. Different subdivisions of characterisation will be covered and the relation in between the transitions are discussed.

Multi–Method Integration. As a substantial last chapter, the aspects of dynamic hybrid models will be generalised in the sense of the continuous dynamics description. Up to this chapter only DAE models are covered for the different states of the hybrid model. In a mathematical modelling sense, this restriction is not always convenient. The integration of other methods to describe the dynamic in one of the involved model description, leads to the last generalisation in this thesis; the integration of several modelling methods in the hybrid model.

Chapter 2

Mathematical Framework for Dynamic Hybrid Models

The first chapter is about the basic definition of models. In the thesis the terms model and system will be used as synonymous. The term model shows more the origin in the surrounding of mathematical modelling, the term system implies the connection to system theory. Due to these two origins the first chapter will introduce as well the term hybrid model in two steps. The first definition will be given in an abstract mathematical setup, the second part will focus on the system simulation approach.

The mathematical definition will combine a discrete structure, represented by an automaton, to a continuous dynamic behavior, represented by ordinary differential equations (ODE) or differential algebraic equations (DAE). The definitions will enable a mathematical understanding of the term dynamic hybrid model.

Following this section the definition will be reformulated for system simulation environments. The focus there, is more oriented on the dynamic system specification, which will be combined with the discrete structure of a hybrid model. The combination will result in the same representation with a different formalism.

To finish this chapter, a selection of examples for dynamic hybrid systems will be given. The examples allow a closer look on the application of the mathematical description of this chapter.

2.1 Mathematical Definition of Dynamic Hybrid Models

This section introduces the notation and basic mathematical definition in the field of hybrid models, respectively hybrid systems. The following subsections will setup the term hybrid system step by step and will also collect related definition of system structures in these field.

2.1.1 Dynamic Systems

The classical definition of dynamic systems ist possible in two ways. One possibility deals with the definition by ordinary differential equations which is the standard approach for continuous system theory. Another way uses a more abstract definition which opens a possibility for more generality regarding the definition. Nevertheless for proper constraints, the connection to the classical definition in continuous system theory is given.

According to [33] the definition of a dynamic system is given by the following structure.

Definition 2.1. Let T and $X \neq \emptyset$ denote sets, with $T \in \{\mathbb{N}_0, \mathbb{Z}, \mathbb{R}_0^+, \mathbb{R}\}$, X an arbitrary topological space, and $g\colon T \times X \to X$ a mapping, which fulfill for $x \in X$ and $t_1, t_2 \in T$

(a) $g(0, x) = x$,

(b) $g\big(t_1, g(t_2, x)\big) = g(t_1 + t_2, x)$.

The triple (T, X, g) is called *dynamic system* with the addendum for $T \in \{\mathbb{N}_0, \mathbb{Z}\}$ *time–discrete* and for $T \in \{\mathbb{R}_0, \mathbb{R}\}$ *time–continuous.* X is called the *state space* and g the *flow*. To provide a tight notation related to t and x the notation of the mapping g is adapted to

$$g_t(x) := g(t, x).$$

Remark 2.2. Some remarks to the definition of a dynamic system:

(a) In most of the application examples, the state space will be a structure equal or isomorphic to $\mathbb{R}^n$.

(b) For classical system theory the mapping g will be in the form of $g\colon \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ or $g\colon \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^+$.

(c) Because of the restriction $T \in \{\mathbb{N}_0, \mathbb{Z}, \mathbb{R}_0^+, \mathbb{R}\}$ it is assumed the existence of an operation $+\colon T \times T \to T$ to formulate the requirement 2.1 (b) for the mapping g. This requirement is called the *monoid property* remembering on the algebraic attribute.

Theorem 2.3. Let $+\colon T \times T \to T$ be a commutative binary operation on T. It apply for $t, t_1, t_2 \in T$

(a) $g_0 = \mathrm{id}$,

(b) $g_{t_1+t_2} = g_{t_1} \circ g_{t_2} = g_{t_2} \circ g_{t_1}$,

(c) $g_t^{-1} = g_{-t}$.

Proof.

$$g_t\big(g_{-t}(x_0)\big) = g_{-t}\big(g_t(x_0)\big) = g_0(x_0) = \big(g_t^{-1} \circ g_t\big)(x_0) = \big(g_t \circ g_t^{-1}\big)(x_0).$$

□

As mentioned in the beginning the definition of a dynamic system, as given above, allows under certain conditions to illustrate the link to the classical system theory where the definition is given via ordinary differential equations.

Theorem 2.4. Assume $\boldsymbol{x}\colon\ \mathbb{R} \to \mathbb{R}^n$, $\boldsymbol{x} \in \mathcal{C}^1(\mathbb{R})$ and g continuously differentiable. For time–continuous dynamic systems Definition 2.1 leads to

$$\boldsymbol{x}'(t) = f\big(\boldsymbol{x}(t)\big), \qquad \text{with} \qquad f\big(\boldsymbol{x}(t)\big) = \left(\frac{\partial}{\partial t} g_t \bigg|_{t=0} \right)\big(\boldsymbol{x}(t)\big).$$

Proof. With $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\boldsymbol{x}(t) = g_t(\boldsymbol{x}_0)$ consider

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) &= \lim_{h\to 0} \frac{1}{h}\big(g_{t+h}(\boldsymbol{x}_0) - g_t(\boldsymbol{x}_0)\big) = \left(\left(\lim_{h\to 0} \frac{1}{h}(g_h - \mathrm{id}) \right) \circ g_t \right)(\boldsymbol{x}_0) = \\ &= \left(\frac{\partial}{\partial t} g_t \bigg|_{t=0} \circ g_t \right)(\boldsymbol{x}_0) = \left(\frac{\partial}{\partial t} g_t \bigg|_{t=0} \right)\big(\boldsymbol{x}(t)\big). \end{aligned}$$

□

Remark 2.5. Because of the relation shown in Theorem 2.4, the mapping g is called the flow of the differential equation system.

2.1.2 Time–continuous State Space Model

Definition 2.6. Time–continuous State Space Model $\boldsymbol{x}\colon\ \mathbb{R} \to \mathbb{R}^n$ denote the *state space vector* $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\mathbb{R}^n$ is called the *state space* and $\boldsymbol{w}\colon\ \mathbb{R} \to \mathbb{R}^q$ the *vector of external variables*. Both vectors are related by a mixed set of differential and algebraic equations represented with the function $F\colon\ \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^m$ by

$$F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}, t) = 0. \tag{2.1}$$

Solutions of (2.1) are all sufficiently smooth time functions $\boldsymbol{x}$ and $\boldsymbol{w}$ which are satisfying the differential algebraic equation (2.1).

If F is continuous differentiable and $F_{\dot{\boldsymbol{x}}}$ is regular $\dot{\boldsymbol{x}}$ can be expressed according to the implicit function theorem as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{w}, t),$$

for a certain function $f\colon\ \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^n$. Furthermore it is reasonable to have a closer look on the vector of external variables. In technical applications the formulation of a problem asks for a *input–state–output* characterisation of the system to form interfaces for interaction with the surrounding, as well as for combination and collaboration of different systems. This idea leads from the dynamic system with one state vector and one external vector, to an input–output behaviour as shown in Figure 2.1.

Definition 2.7. The vector $\boldsymbol{w} \in \mathbb{R}^q$ is split into a vector $\boldsymbol{u} \in \mathbb{R}^m$ and a vector $\boldsymbol{y} \in \mathbb{R}^p$, restricted by the condition $m + p = q$. $\boldsymbol{u}$ is called the *input* vector or input variables and $\boldsymbol{y}$ the *output* vector or the output variables. Such an input–state–output system is defined via two functions $f\colon\ \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$, $h\colon\ \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^p$, the set of equations

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, \boldsymbol{u}, t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y} &= h(\boldsymbol{x}, \boldsymbol{u}, t), \end{aligned} \tag{2.2}$$

and is called a *State Space Model*.

State space models are important for system simulation approaches, which will be introduced in section 2.2. Due to the need of this definition in this first section a small overlap is visible. The input–output structure of such a state space model is shown in Figure 2.1.

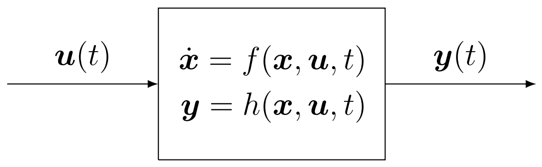

$\boldsymbol{u}(t)$
$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t)$
$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, t)$
$\boldsymbol{y}(t)$

Figure 2.1: Time–continuous Input–Output State Space Model.

In this definition of a State Space Model of course the still important *linear State Space Model* of type

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A\ \boldsymbol{x}(t) + B\ \boldsymbol{u}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= C\ \boldsymbol{x}(t) + D\ \boldsymbol{u}(t), \end{aligned} \tag{2.3}$$

for $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$ and $D \in \mathbb{R}^{p \times m}$, is respected.

2.1.3 Finite Automata

Definition 2.8. A *finite automaton* is described by a triple (L, A, E). L is a finite set called the *state space*, A is a finite set called the *alphabet* whose elements are called *symbols*. E is the transition rule represented as a subset of $L \times A \times L$ and its elements are called *edges*, *transitions* or *events*. A sequence $(l_0, a_0, l_1, a_1, \ldots, l_{n-1}, a_{n-1}, l_n)$ with $(l_i, a_i, l_{i+1}) \in E$, for $i = 0, 1, \ldots, n-1$, is called a *trajectory* or *path*.

The definition of a finite automaton requires to include the explicit specification of a subset $I \subset L$ of *initial states* and a subset $F \subset L$ of *final states*. A path $(l_0, a_0, l_1, a_1, \ldots, l_{n-1}, a_{n-1}, l_n)$ is called a *successful path* if the condition $l_0 \in I$ and $l_n \in F$ is added.

Typical finite automata are represented by a finite graphs. A certain example and the illustration is shown in the following.

Example 2.9. Consider $L = \{l_0, l_1, l_2, l_3\}$ and the alphabet $A = \{a_0, a_1, a_2, a_3\}$. A can be interpreted as label on the edges of the illustration of the finite graph and the elements of L are denoting the nodes of this graph. The illustration of this graph is shown in Figure 2.2.

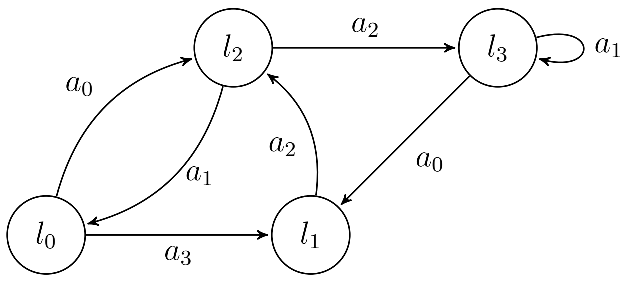


Figure 2.2: Example of a Finite Graph as an Illustration for a Finite Automaton.

Also in the context of finite automata exists an input–output automaton. Therefor two symbols, namely an *input* symbol i and an *output* symbol o, has to be chosen in the finite automaton. Deterministic input–output automata can be represented by equations

$$l^\sharp = \nu(l, i) \qquad \text{and} \qquad o = \eta(l, i). \tag{2.4}$$

The symbol $l^\sharp$ in equation (2.4) denotes the new value of the state after the event takes place, resulting from the old discrete state value l and the input i. An alternative formulation in this situation could be given: If t_k is the time step before the event takes place and t_{k+1} the time step after the event the relations for the output state can be considered as $l = l(t_k)$ and $l^\sharp = l(t_{k+1})$. Thereby $l\colon \mathbb{Z} \to L$ is the time evolution of l.

In contrast to the continuous–time systems, the solution concept of a finite automaton with or without initial and final states is completely specified. The behavior of the finite automaton consists of all paths respectively successful paths of the automaton.

2.1.4 Hybrid Automata

The combination of the previous definitions of a continuous–time state space system in Definition 2.6 and a finite automaton in Definition 2.8, leads to the definition of a hybrid automaton an thereby to the mathematical definition of a hybrid model.

Definition 2.10 (Hybrid Automaton)**.** For the environment it is defined:

(a) L is a finite set, called the *set of discrete states*.

(b) X is the *continuous state space* of the hybrid automaton in which the *continuous state variables*, elements of the vector x, take their values. Typically X fulfills $X \subseteq \mathbb{R}^n$ and $\boldsymbol{x} \in X$ is the *continuous state vector*, but also n–dimensional manifolds are imaginable.

(c) A is a finite *set of symbols* which labels the edges.

(d) $W = \mathbb{R}^q$ is the *continuous communication space* in which the *continuous external variables* of the vector $\boldsymbol{w} \in W$ take their values.

(e) E is a finite set of edges, called *transitions* or *events*. Every edge is defined by a tuple $(l, a, \mathrm{Guard}_{l,l'}, \mathrm{Jump}_{l,l'}, l')$, where $l, l' \in L$, $a \in A$.
$\mathrm{Guard}_{l,l'}$ is a subset of X and $\mathrm{Jump}_{l,l'}$ is a relation defined by a subset of $X \times X$. The transition from the discrete state l to l' is called *enabled*, when the continuous state fulfills $\boldsymbol{x} \in \mathrm{Guard}_{l,l'}$. The continuous state $\boldsymbol{x}$ jumps to a value $\boldsymbol{x}'$ if $(\boldsymbol{x}, \boldsymbol{x}') \in \mathrm{Jump}_{l,l'}$.

(f) $\mathrm{Inv}\colon L \to \mathcal{P}(X)$ is a mapping from the locations L to the set of subsets of X that is $\mathrm{Inv}(l) \subset X$, $\forall l \in L$. Whenever the system is at location l, the continuous state $\boldsymbol{x}$ must satisfy $\boldsymbol{x} \in \mathrm{Inv}(l)$. $\mathrm{Inv}(l)$ for $l \in L$ is called the *location invariant* of location l.

(g) $\mathrm{Act}\colon L \to F_L$ is a mapping that assigns to each location $l \in L$ a set of differential algebraic equations $F_l \in F_L$, relating the continuous state vector $\boldsymbol{x}$ with their time–derivative $\dot{\boldsymbol{x}}$ and the continuous external variables $\boldsymbol{w}$ using $F_l\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^m$ by

$$F_l(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0. \tag{2.5}$$

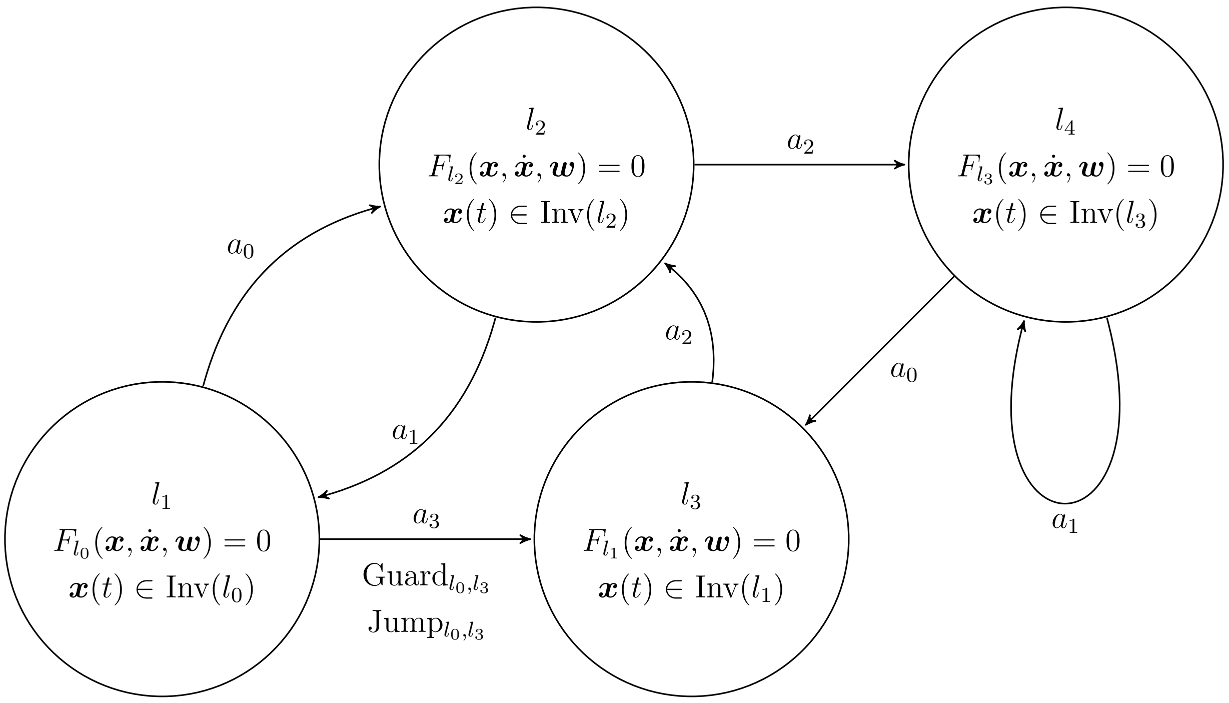
$m \in \mathbb{N}$ is fixed for each location $l \in L$ and the solutions of these differential algebraic equations are called the *activities* of the location l .

The tuple $(L, X, A, W, E, \mathrm{Inv}, \mathrm{Act})$ is called a *hybrid automaton.*

A hybrid automaton can be illustrated in the same way like finite automata, consider the following

Example 2.11. Continuation of Example 2.9. The discrete setup of the automaton stay the same only the framework of the states will be extended by the differential algebraic content. Each node represents a corresponding DAE system given by $F_{l_0}, \dots, F_{l_3}$.

For representation of the hybrid automata again a finite graph is used, see Figure 2.3. To be well–arranged only for one transition (l_0, a_3, l_1) the sets Guard and Jump is illustrated. Moreover each node l_k is defined by the DAE $F_{l_k}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$ and additionally by the statement $\boldsymbol{x}(t) \in \mathrm{Inv}(l_k)$.



l_2
$F_{l_2}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_2)$
a_2
l_4
$F_{l_3}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_3)$
a_0
a_2
a_0
a_1
l_1
$F_{l_0}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_0)$
a_3
Guard_{l_0,l_3}
Jump_{l_0,l_3}
l_3
$F_{l_1}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_1)$
a_1

Figure 2.3: Example of a Finite Graph as an Illustration for a Hybrid Automaton.

Definition 2.12. A *continuous trajectory* $(l, \delta, \boldsymbol{x}, \boldsymbol{w})$ associated with a location $l \in L$ consists of a time $\delta \geq 0$, the duration of the continuous trajectory, a piecewise continuous function $\boldsymbol{w}\colon [0, \delta] \to W$ and a continuous and piecewise differentiable function $\boldsymbol{x}\colon [0, \delta] \to X$ which fulfill

(a) $\boldsymbol{x}(t) \in \mathrm{Inv}(l), \ \forall t \in (0, \delta)$,

(b) $F_l(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{w}(t)) = 0, \quad \forall t \in (0, \delta) \setminus \{\hat{t}_1, \dots, \hat{t}_w\}$, where $\{\hat{t}_1, \dots, \hat{t}_w\}$ is the set of discontinuity of $\boldsymbol{w}$ in $(0, \delta)$.

Definition 2.13. A *trajectory of the hybrid automaton* is an (infinite) sequence of continuous trajectories

$$(l_0, \delta_0, \boldsymbol{x}_0, \boldsymbol{w}_0) \xrightarrow{a_0} (l_1, \delta_1, \boldsymbol{x}_1, \boldsymbol{w}_1) \xrightarrow{a_1} (l_2, \delta_2, \boldsymbol{x}_2, \boldsymbol{w}_2) \xrightarrow{a_2} \dots \tag{2.6}$$

so that at the event times

$$t_0 = \delta_0, \quad t_1 = \delta_0 + \delta_1, \quad t_2 = \delta_0 + \delta_1 + \delta_2, \quad \dots, \tag{2.7}$$

the following inclusions hold for the discrete transitions

$$\begin{gathered} \boldsymbol{x}_j(t_j) \in \mathrm{Guard}_{l_j, l_{j+1}}, \\ (\boldsymbol{x}_j(t_j), \boldsymbol{x}_{j+1}(t_j)) \in \mathrm{Jump}_{l_j, l_{j+1}} \end{gathered} \tag{2.8}$$

for all $j = 0, 1, 2, \dots$.

The definition of a trajectory of the hybrid automaton associate the possibilities of illustrations of trajectories as usual for dynamic systems. In the case of $x \in \mathbb{R}$ the illustration of x, can be illustrated as in Figure 2.4. It represents the consecutively illustration of x of the different states of a hybrid automaton.

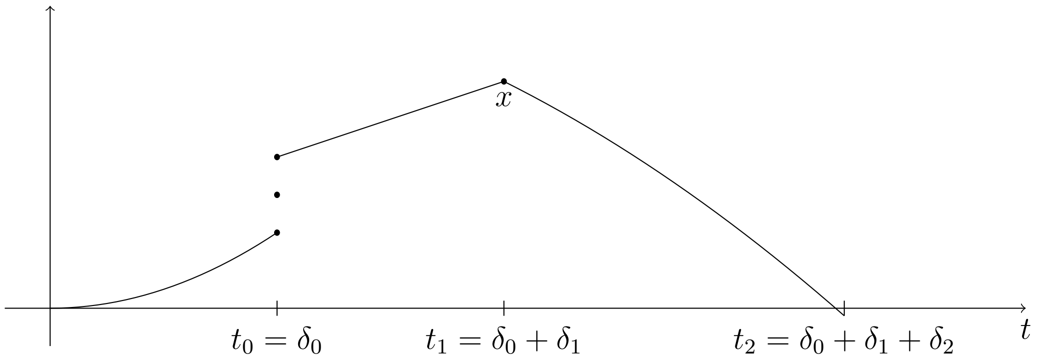


Figure 2.4: An exemplary Illustration of the Time Characteristics of x for a Hybrid Trajectory with four States.

Remark 2.14. Some additional information regarding a trajectory of a hybrid automaton, an illustration of the continuous part of the state of the automaton and some issues regarding the analysis of trajectories of the automaton:

(a) In the environment of a hybrid automaton a *state* is given by $(l, \boldsymbol{x})$ or $(l, \boldsymbol{x}(t))$. This state denotes the discrete state as well as the continuous state. In the field of a mathematical dynamical system $\dot{x} = f(x, t)$ the state would be x and in cases if possible the trajectory x would be drafted. In case of a trajectory of a hybrid automaton, more than the state x has to be considered. But for further examples it will be necessary to sketch the characteristics over time. In this cases the way of notation will be used that the trajectory of the continuous part of the hybrid automaton will be represented.

(b) While studying trajectories of hybrid automaton it could happen after some time, that the system ends in a state $(l^*, \boldsymbol{x}^*)$ from which there is no connection, no possible transition to another location. This situation is called a *deadlock* according to computer science.

(c) In general the duration δ_i of a certain continuous trajectory is allowed to be zero. This allows that at a certain time $t \in \mathbb{R}$ several socalled *timeless events* can take place. This case of *multiple event* has a underlaying time axis which has a more complex structure than $\mathbb{R}$. A certain time instant $t \in \mathbb{R}$ correspond to a list of sequences of transitions which are passed all at the same point in time, see [38, 26].

2.2 System Simulation Description of Dynamic Hybrid Models

2.2.1 Continuous System Theory

In the continuous systems theory one usual description of systems is given by equation (2.2). In this case an explicit functional relation to express $\dot{\boldsymbol{x}}$ and an output function h to express to output vector $\boldsymbol{y}$ is needed. For a given initial value $\boldsymbol{x}_0$ and a fixed input vector $\boldsymbol{u}$

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t), \\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{cases} \tag{2.9}$$

describes an initial value problem. For this the question of existence and uniqueness of solution can be answered by the Picard–Lindelöf theorem, listed for example in [27]. The difference in the formulation for system theory and dynamical system theory in mathematics is the missing input. In mathematical circumstances a dynamical system is formulated as $\dot{x} = f(t, x)$, in system theory also the input has to be considered. The input is known for different problems it has no influence on the theorem, only the notation has to be adopted.

Theorem 2.15 (Picard–Lindelöf)**.** If $f\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ fulfills for a given $\boldsymbol{x}_0$, a fixed $\boldsymbol{u}$ and $L \in \mathbb{R}$ the inequality (Lipschitz condition)

$$\|f(\boldsymbol{x}, \boldsymbol{u}, t) - f(\hat{\boldsymbol{x}}, \boldsymbol{u}, t)\|_2 \leq L\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2$$

for all $\boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^n$ and $t \in \mathbb{R}$ than a unique solution of the initial value problem (2.9) exists.

Remark 2.16. Some additional remarks to the formulation of the initial value problem:

(a) Due to the focus of the thesis in Theorem 2.15 the Euclidean norm is used. The theorem of course is available in a more general formulation in Banach spaces.

(b) In the field of system theory the input–output formulation is often used. For this reason in the initial value problem the input vector $\boldsymbol{u}$ appears. For classical dynamical systems this input vector does not exist. Due to the character of an input vector it can be assumed as fixed values and so it is not influencing the theorem.

In several questions related to subjects of technical and natural sciences, this model description is not enough. Not only differential dependencies are observed but as well restrictions or balance equations are of interest. To create a more general modelling approach the system (2.9) will be completed by further variables as $\boldsymbol{z} \in \mathbb{R}^\ell$, represents the *vector of algebraic* *variables* which are not included in the state vector, and $\boldsymbol{p} \in \mathbb{R}^r$ the *vector of parameters* which includes inter alia the initial condition $\boldsymbol{x}_0$. Moreover, in order to respect any additional algebraic conditions in the model an algebraic equation is added which is defined via the function $g\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^u$ with

$$g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}.$$

This generalisation results in a model description which is given by a *differential algebraic equation* of the form

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}. \end{cases} \tag{2.10}$$

To soften the restriction of the given differential equation in an explicit form with respect to $\dot{\boldsymbol{x}}$ the description can be replaced by an implicit differential algebraic equation. For this purpose assume a function $F\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^s$ which define the *implicit differential algebraic equation* via

$$F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}. \tag{2.11}$$

Note 2.17. Some notices with respect to notation:

(a) As mentioned in the text above, the extension of the model description to a model which includes algebraic variables extends also the differential part of the description. Even though this function f is different to the one used above, the letter f has not changed. This increases the readableness of the models in this thesis, where f represents always the dynamic, being the right hand side of $\dot{\boldsymbol{x}}$.

(b) It is easy to show that the implicit DAE formulation is more general. The explicit formulation can be written as $\dot{\boldsymbol{x}} - f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = 0$ and combined to the algebraic equation. Vice versa in general it can not be guaranteed that the implicit function can be solved with respect to $\dot{\boldsymbol{x}}$, therefore, the implicit function theorem has to be consulted.

Modelling in continuous system theory mainly focuses on an input–output description. This means that the model description in an implicit differential algebraic equation formalism for the dynamics of the system has to be combined with an output equation similar to (2.2) extended by algebraic variables and parameter. This will be summarised in the following

Definition 2.18 (Input–Output DAE Model)**.** Assume two functions

$$F\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^s \quad \text{and} \quad h\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^p\,.$$

An *implicit differential algebraic input–output model* is given in the form

$$\begin{cases} F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases} \tag{2.12}$$

The general definition for the input–output DAE model respects as well the linear state space model in a trivial sense. Equation (2.3) can be formed to an implicit ordinary differential equation like it was mentioned in Remark 2.17. The algebraic equation simple does not exist and the output equation is used as it is given. This shows the more general modelling purpose.

Another topic to address is the definition of such an input–output DAE model. The general Definition 2.18 include no restrictions, the domains of F and h have no restrictions. In applications this is rather extraordinary. In usual application examples, certain restrictions regarding the domain of definition of the functions F and h are provided. At least restrictions with respect to time, if causal systems are investigated, are present. Also restrictions regarding the image of $\boldsymbol{x}$, similar to $\boldsymbol{x}(t) \in \mathrm{Inv}(l)$ in the definition of a hybrid automaton, are present. This leads to the extension of the classical continuous system theory to the so called state space modelling which realise hybrid behaviour in the setup of system theory.

2.2.2 State Event Modelling

The classical continuous system theory does not allow discontinuities in the right hand side of a dynamical system introduced in (2.9). Such a discontinuity violates the theorem of Picard–Lindelöff which guarantees existence of solution x. In applications discontinues changes are a basic need in modelling theory for several purposes. Some simple examples are ideal valves in fluid dynamics, diodes in electrical circuits or switching elements in different fields of application. This rather simple elements can violate the continues description of system theory but this elements are often used in application. State event modelling offers a method to model this discontinuities by using piecewise system theory.

As a motivation for state event modelling let's assume the following

Example 2.19 (Saturation Effect)**.** Consider the simple linear state space description

$$\begin{aligned} \dot{x}(t) &= x(t) + u(t), \\ y(t) &= x(t) - u(t), \end{aligned}$$

for input, output and state vectors $u, x, y\colon [0,\infty) \to \mathbb{R}$. The initial condition is given by $x(0) = 0$.

Assume an additional condition which address a bound to the state variable x. The additional condition with respect to x turn out the model description

$$\left.\begin{aligned} \dot{x}(t) &= x(t) + u(t) \\ y(t) &= x(t) - u(t) \end{aligned}\right. \quad \text{for } x \le p_1, \qquad \text{and} \qquad y(t) = p_2 \quad \text{for } x > p_1,$$

where $p_1 = \mathrm{e}^3 - 4$ and $p_2 = \mathrm{e}^3 - 7$. Its obvious that the determination whether which part of the model is valid, is done via the condition to the state vector x.

Due to the simple underlaying structure of the linear model in this case, for a given input vector, the analytical solution can be derived. Let's consider a ramp as an input signal, this means $u\colon [0,\infty) \to \mathbb{R}$ is given by $u(t) = t$ and so $\dot{u}\colon (0,\infty) \to \mathbb{R}$ results in $\dot{u}(t) = 1$.

Either the differential equation for the state $\dot{x} = x + t$ can be solved, or computed from the differential equation system the input–output characterisation as

$$\dot{y} - y = 2t - 1.$$

The solutions of x and y is

$$x(t) = \mathrm{e}^t - t - 1 \qquad \text{and} \qquad y(t) = \mathrm{e}^t - 2t - 1.$$

Both satisfy the output equation $y(t) = x(t) - u(t)$.

Due to the analytical solutions it is possible to find out the time t^*, where the condition regarding x switches between the two specifications of the model. The separation is done by $x \leq p_1$ which gives, using the analytical solution, the equation $\mathrm{e}^t - t - 1 = \mathrm{e}^3 - 4$ and so $t^* = 3$. It is obvious that for $t = 3$, y results exactly in the given parameter $p_2 = y(3) = \mathrm{e}^3 - 7$.

The input–output behaviour can be summarised in a common characteristics over time, illustrated in Figure 2.5.

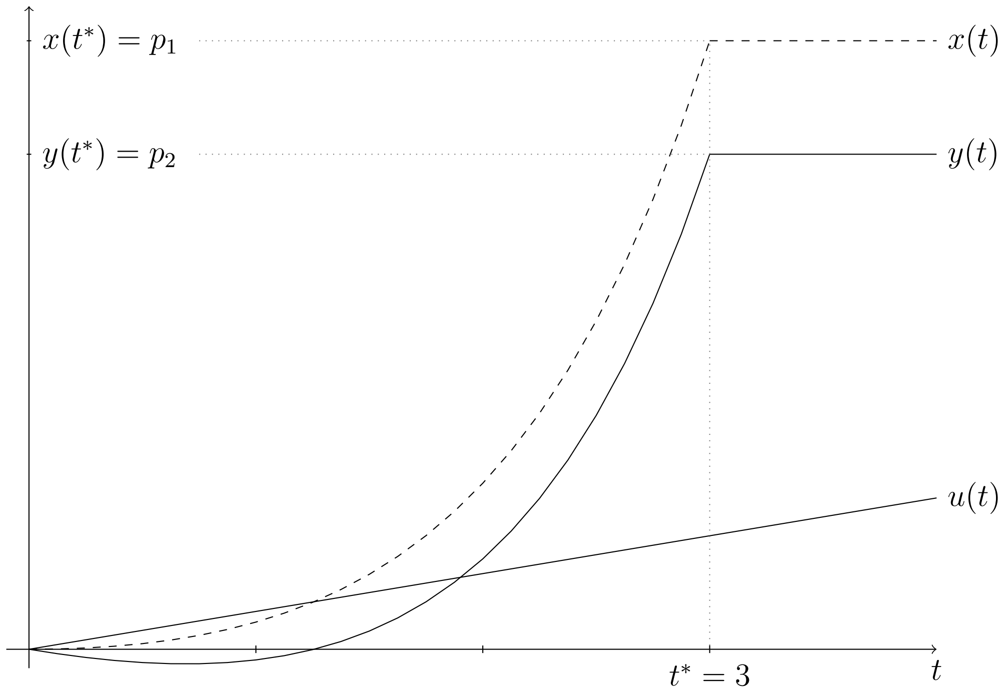


Figure 2.5: Illustration of the Input u, State x and Output y Vectors of the Saturation Model in Example 2.19.

The basic function of this example is to model a saturation effect. Starting from a certain value an increasing input results in an output which is not following this behaviour, it is fixed on a certain level. This effect can be observed in electric amplifier, which saturate after a certain input value. The saturation effect is usually defined by a level in the input vector, due to the analytical relationship it was possible to formulate this bound also for the state vector.

Example 2.19 introduce a simple scenario where conditions regarding the state vector are used to implement a model with different regions for the model description. The analytical solutions made it as well possible to determine the time point where the condition w.r.t. the state vector was fulfilled. Another scenario can direct the conditions not to the state vector but the condition is formulated w.r.t. a certain moment. The following example will give a motivation for this time driven condition.

Example 2.20 (Discontinuity in Model Description)**.** Assume a simplified state space description in the form

$$\begin{aligned}\dot{x}(t) &= f(t)x(t) + g(t), \quad x(0) = x_0\\ y(t) &= f(t)x(t) + g(t),\end{aligned}$$

where x and y are the state and the output vector $x, y\colon [0,\infty) \to \mathbb{R}$ and g is a (generalised) input function $g\colon [0,\infty) \to \mathbb{R}$. The specialty in this model is the time dependent coefficient $f\colon [0,\infty) \to \mathbb{R}$ which is not continuous and a analytical solution is not to be expected. At first assume f as a shifted Heaviside function

$$f(t) = \mathrm{H}(t - t_0) = \begin{cases} 0, & t \le t_0,\\ 1, & t > t_0.\end{cases}$$

Consider for this example $g(t) = 2t - 1$ as a generalised input, $t_0 = 2$ for the shifted Heaviside function and $x_0 = 1$ as an initial condition. In this case the whole model can be split to

$$\begin{aligned}\dot{x}(t) &= g(t),\\ y(t) &= g(t),\end{aligned} \quad \text{for} \quad t \le 2 \qquad \text{and} \qquad \begin{aligned}\dot{x}(t) &= x(t) + g(t),\\ y(t) &= x(t) + g(t),\end{aligned} \quad \text{for} \quad t > 2.$$

Due to the linear model structure in this situation analytical solutions are as well available. For $t \le 2$ the system is a simple input through system $y(t) = g(t)$ and the solution is given by

$$x(t) = t^2 - t + 1 \qquad \text{and} \qquad y(t) = 2t - 1.$$

For $t > 2$ the system differential equation is given by $\dot{x}(t) - x(t) = 2t - 1$. This yield

$$x(t) = 2\mathrm{e}^t - 2t - 1 \qquad \text{and} \qquad y(t) = 2\mathrm{e}^t - 2.$$

The behaviour in this scenario is separated only by a point in time, in this case defined by the switching point of the Heaviside function. To summarise the input–output behaviour, a characteristics over time is illustrated in Figure 2.6.

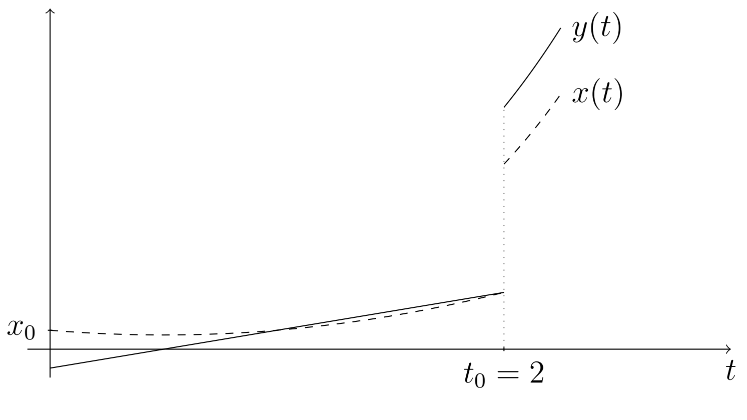


Figure 2.6: Illustration of the State x and Output y Vectors of the Discontinuity in Model Description by Heaviside Function in Example 2.20.

In a second step f is considered as a shifted signum function, given by

$$f(t) = \operatorname{sign}(t - t_0) = \begin{cases} -1, & t < t_0, \\ 0, & t = t_0, \\ 1, & t > t_0. \end{cases}$$

Assume still $g(t) = 2t - 1$ as a generalised input, $t_0 = 2$ for the shifted signum function and $x_0 = 1$ as an initial condition. Under this condition the model can be formulated as

$$\begin{aligned} \dot{x}(t) &= -x(t) + g(t), \\ y(t) &= -x(t) + g(t), \end{aligned} \quad \text{for} \quad t < 2, \qquad \begin{aligned} \dot{x}(t) &= g(t), \\ y(t) &= g(t), \end{aligned} \quad \text{for} \quad t = 2$$

and

$$\begin{aligned} \dot{x}(t) &= x(t) + g(t), \\ y(t) &= x(t) + g(t), \end{aligned} \quad \text{for} \quad t > 2.$$

This model is characterised through three different dynamical systems, where the solutions can be given analytically. For $t < 2$ the solution for x and y are given by

$$x(t) = 2\mathrm{e}^{-t} - 2t - 1 \qquad \text{and} \qquad y(t) = 4t - 2\mathrm{e}^{-t},$$

for $t = 2$ by

$$x(t = 2) = t^2 - t + 1\big|_{t=2} = 3 \qquad \text{and} \qquad y(t = 2) = 2t - 1\big|_{t=2} = 3$$

as well as for $t > 2$ by

$$x(t) = 2\mathrm{e}^{t} - 2t - 1 \qquad \text{and} \qquad y(t) = 2\mathrm{e}^{t} - 2.$$

The signum function switch over the model into three different modes. The analytical solutions are illustrated in Figure 2.7.

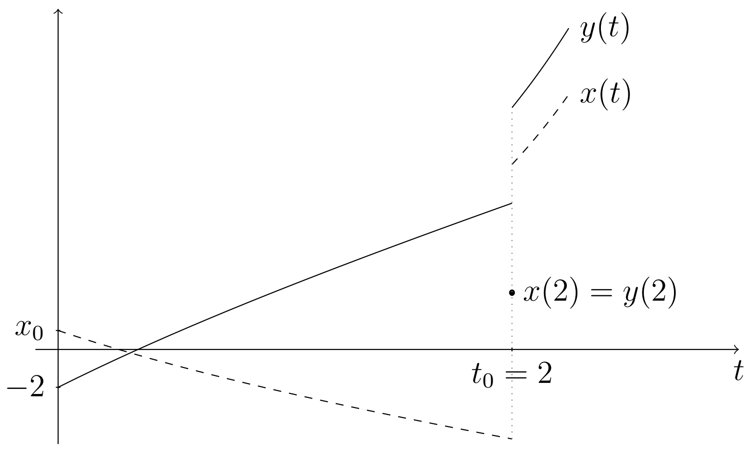


Figure 2.7: Illustration of the State x and Output y Vectors of the Discontinuity in Model Description by Signum Function in Example 2.20.

This example shows a discontinuous change in the model description which results in a condition w.r.t. time. Obviously this is different to a condition which involve the state vector x as it was shown in Example 2.19. These two structurally different conditions resulte in a different form of description and in a different handling of the model transition. According to this two motivating examples it is reasonable to define a term event to point out that at this fulfilled condition the model is structural changing and the distinction between a condition w.r.t. time or a state value.

Definition 2.21. The discontinuous changes of a model description, in general given by Definition 2.18, are called *events*.

(a) If the time instant of the change is known in advance, the event is called a *time event*.

(b) If the event depends on a certain value of the state variable or functions of state variables the event is called a *state event*.

Referring to this definition, Example 2.19 and 2.20, the introducing examples are representing a sate event in the case of the saturation and a time event in the case of the discontinuity in the model description.

The next question deals with the relation of both events to each other. A closer look to the solutions of Example 2.19 shows, that the state event appears at a certain moment t^*. The resulting question is if there are some equivalences between this two definitions and are there some conditions where both definitions are well separated to each other.

The answer to the question of the equivalences is related to the analytical solution of the model in Example 2.19. The specification is such designed as the equation $x(t) = p_1$ has one unique solution. In this case it is possible to determine the time t^* where the condition w.r.t. the state value is fulfilled. If it is not possible to derive the unique time analytically an additive numerical algorithm, to find the numerical approximation, is used. In this case the relevance of a state event is more clear because it can not be transformed in an equivalent condition w.r.t. time. It is obvious that a time event is easier to handle than a state event, because the mentioned additional computations are not necessary.

Coming back to the mathematical formulation, a functional definition of the event must be given.

Definition 2.22 (Event Function)**.** Consider the model characterisation given in Definition 2.18. If a function $e\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \to \mathbb{R}^\nu$, which is called *event function*, whose zeros $t^* \in \mathbb{R}$ of

$$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \mathbf{0}$$

determines the time instant of the occurrence of an event, a *state event* is defined. Otherwise, if the event function is structured as $e\colon \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^\nu$ and the zeros $t^* \in \mathbb{R}$ of

$$e(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \mathbf{0}$$

determines the time instant, a *time event* is defined.

In general, the event function and hence in the definition of an event include the input, the parameter and the algebraic variables. The characterisation of a time or state event is separated by the depending variables of e. If e depends on the state $\boldsymbol{x}$, a state event is defined. If e depends not on x at all and only on t, a time event is defined. If it is possible to express t analytically from this equation the exact time is known, according to the first announced definition of a time event. This considerations are related to the numerical aspects of locate the event time, see chapter 4. If the event function e allows to solve (numerically) $\boldsymbol{x}$ and in a second step the event time, the event depends on the state. If it is possible to solve (numerically) the event time directly from the event function e, the event does not depend on the state.

Remark 2.23. Some procedural notes regarding the difference and the commonalities of time and state events are given in the following list:

(a) In the cases of a state event, the event function e defines the state event by involving x in the equation $e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \boldsymbol{0}$. Starting from this equation, x has to be derived and after that t has to be computed. This means the time of the event is implicit given via the state x, which explains the name of this event.

(b) Assuming a time event, the event is defined by $e(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}$ without involving the state x. From this equation, the zeros t of the equation has to be solved directly and so they are, so called, explicit given by this equation.

(c) The calculations, nearly in all situations, has to be done numerically because the model structures usually do not allow to compute an analytical solution. Closer investigations on this aspect will be done in subsection 2.2.3.

(d) For both types of events, the event time is important to know to perform the event in the model structure. This issue is as well addressed in the subsection 2.2.3 which deals with numerical aspects.

(e) In simple or academic examples it can happen that x is analytically available, as it happened in Example 2.19. In this case it was possible to insert the analytical solution of x and transform the state event in a time event. This means the complexity of the model itself controls if a state event or time event can be formulated. But this question belongs to the modelling process itself and not to the characterisation addressed in this work.

Theorem 2.24 (Equivalence of state and time event)**.** Consider a model characterisation given in Definition 2.18, an event function $e\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \to \mathbb{R}^\nu$ and a state event defined with this function via

$$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \boldsymbol{0}.$$

The state event is equivalent to a time event, if, and only if the analytical solution $x\colon \mathbb{R} \to \mathbb{R}^n$ is known.

Proof. At this point of consideration assume the input $\boldsymbol{u}$, the algebraic variables $\boldsymbol{z}$ and the parameter vector $\boldsymbol{p}$ as constant. Furthermore the analytical solution of $x\colon \mathbb{R} \to \mathbb{R}^n$ is explicit known, i.e. $\boldsymbol{x} = \boldsymbol{x}(t)$ is a functional relation. Applying this in the state event equation

$$e(\boldsymbol{x} = \boldsymbol{x}(t), \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \overline{e}(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}$$

the equation transforms to a time event defined by the event function $\overline{e}\colon \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^\nu$. $\square$

Equipped with this definitions reconsider Example 2.19 and 2.20 to continue the analysis.

Example 2.25. Consider the specification of Example 2.19. The conditions regarding the separation of the model parts from each other is given by $x \leq p_1$ and $x > p_1$, where $p_1 \in \mathbb{R}$ was a fixed parameter. To formulate this condition via an event function consider

$$e(x, p_1) = x(t) - p_1,$$

with $e\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. In this case the function e is independent from the second parameter p_2 which is present in the model, as well as from the input u and an algebraic variable which does not existent at all in this example. The analytical solution for the model is available according to

$$x(t) = x(t) = \mathrm{e}^t - t - 1,$$

and the parameter value $p_1 = \mathrm{e}^3 - 4$ allows a unique solution for the zeros of the equation

$$e(x, p_1) = x(t) - p_1 = \mathrm{e}^t - t - 1 - (\mathrm{e}^3 - 4) = 0.$$

This means the event can be formulated by using the event function

$$\overline{e}(p_1, t) = \mathrm{e}^t - t - \mathrm{e}^3 + 3 = 0,$$

where the unique solution is given by $t^* = 3$.

This example is an application of Theorem 2.24. x is analytically known, so the state event defined by e can be transformed to a time event represented by $\overline{e}$. x is a monotonically increasing function on $\mathbb{R}$, so it is bijective. This allows to determine the one and only solution for the event time $t^* = 3$.

This continuation of Example 2.19, modelling a saturation effect, shows the relation between a state event and a time event. The second example was the model of a discontinuity, which was given by a time event.

Example 2.26. Continue with the same description as given in Example 2.20. In all considered cases, the Heaviside function on the one hand and the signum function on the other hand, the event was always given via a concrete moment t_0.
In all cases the event can be formulated with one event function $e\colon \mathbb{R} \to \mathbb{R}$ via the equation

$$e(p, t) = t - t_0 = 0.$$

In this model the specified moment t_0 is as well a parameter $p = t_0$. So, the event function e depends on the parameter and time. Again there is no algebraic variable existent and the input is not influencing the event.

The definitions in this subsection allows to understand the approach of state event modelling in the context of continuous system theory. What is missing, is the procedural view for simulation. In general, the task is to use this modelling approach to implement a model in a simulation environment, so the procedure to execute such models has to be discussed. The question how to execute such events is called the handling of state events.

2.2.3 State Event Procedure

State event models consists of two main parts:

(a) model description

(b) state event description

The model description is given by the implicit differential algebraic input–output model

$$\begin{cases} F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \end{cases}$$

the state event description via the event function e and

$$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \boldsymbol{0}.$$

During the further procedure, only state events will be considered because a time event is included in a more simple special case.

For the procedural description of the state event handling, a simulation environment has to be assumed. The simulation environment mainly deals with the mathematical model in an numerical simulation and beside within the simulation, to check if there is an event incidence in the simulation. An overview is illustrated in Figure 2.8. It is pointed out that, the so called state equation delivers the current value of the state vector $\boldsymbol{x}$ to use this value in the computation of the event function.

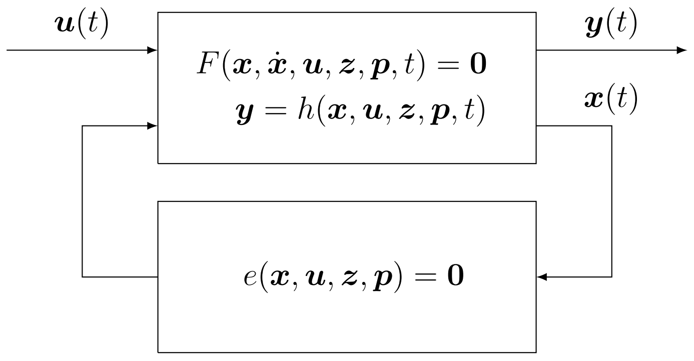

$\boldsymbol{u}(t)$
$F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}$
$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t)$
$\boldsymbol{y}(t)$
$\boldsymbol{x}(t)$
$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \boldsymbol{0}$

Figure 2.8: Illustration of a Simulation run with State Event Detection.

About the aspects of this numerical simulation run chapter 4 will present more details. In this subsection the procedural steps of the state event handling are in the foreground.

Starting from a currently active simulation run, as presented in [34], a state event algorithm requires the following steps:

(a) Detection of an event

(b) Localisation of the event time

(c) Stopping the simulation run

(d) Executing the event action

(e) Adjust initial condition if necessary

(f) Restart the simulation

The localisation of the event time is a numerical duty of the simulation environment. The detection of an event is more an analytical duty which applies theoretical theorems for practical usage. For reasons concerning the usability, normally the event function e is assumed to be at least continuous, sometimes even differentiable.

Assuming an event function e which is continuous w.r.t. time and observe this in an interval $[a, b]$ for $a < b$. The event function is not explicit dependent on t, because a state event is assumed. As discussed before, the event time has to be computed even if the event function is not dependent explicitly on the time. This is numerically done, so a certain series of values of e at a certain point in time t are available.

For reasons of notation e_i is denoted as the value of the event function at a certain time t_i for $i = 1, 2, \dots, n$. In Figure 2.9 an arbitrary continuous event function e is depicted over the time interval $[t_1, t_3]$. The event function is selected in that sense, that in this time interval two zeros are located. The three time points t_i are corresponding to the values of the event function e_i. In this situation the intermediate value theorem can be applied. The signs of the values e_i and e_{i+1} for $i = 1, 2$ are different and the event function is assumed to be continuous. According to the intermediate value theorem respectivley the theorem of Bolzano, all values between e_i and e_{i+1} are in the image of e over the interval $[t_i, t_{i+1}]$, in particular the zero of the event function.

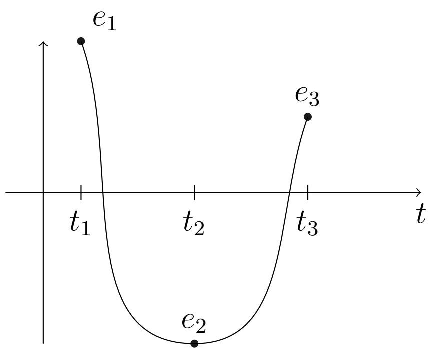


Figure 2.9: An exemplary Time Characteristics of an Event Function e.

Remark 2.27. In the existent version of the formulation of state event handling, the detection of events, assume a function $e = e(t)$. Especially in this case the intermediate value theorem can be applied. Due to the structure of the state events this is not given. Assuming an analytical expression for the solution of the state vector $\boldsymbol{x}$, the input vector $\boldsymbol{u}$ and the algebraic vector $\boldsymbol{z}$ the event function could be derived as a function depending on t. In the current formulation of the event detection, this has to be taken in account to get the idea of the procedure. In the field of modelling and simulation this handling strategy will be applied in a numerical approach, where the addressed values e_i will be the numerical approximation of the values of the event function.

A closer look to this state event location and linked to this topic the state event handling will be addressed in the Chapter 4 State Event Handling in the Numerical Environment. There will be given a closer look to the numerical algorithms and possibilities in this field.

2.3 Examples of Hybrid Models

This section is dedicated to give some introductory academic examples for hybrid automata on the one hand and some state event models on the other hand. Particular examples will be introduced in both environments to show the similarities and the differences of both mathematical environments.

2.3.1 Control System with a Hysteresis Element

The example is introduced via a control system with a *hysteresis element* in the feedback loop, as suggested in [6] and [50]. The hysteresis element has the certain characteristics which is sensitive regarding the direction of running through the characteristics. To have an impression about this sensitiveness, the characteristic of H is sketched in Figure 2.10.

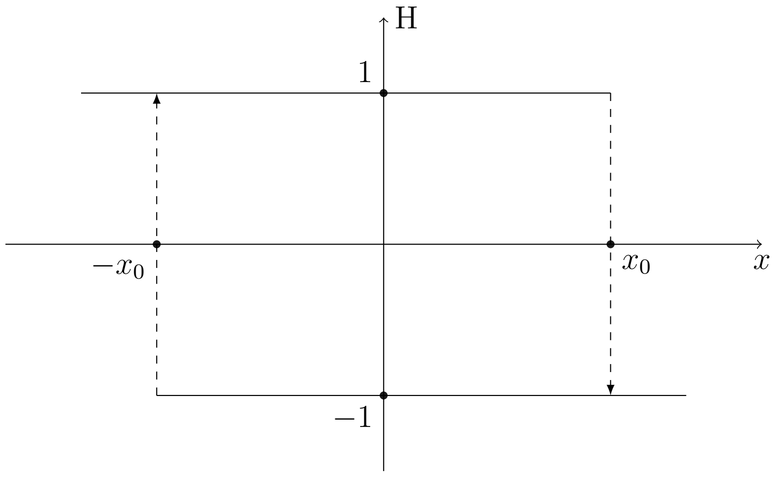


Figure 2.10: The Characteristics of a Hysteresis Element H.

The control system is specified by using the hysteresis element and the equation

$$\dot{x} = \mathrm{H}(x) + u. \tag{2.13}$$

If the description of the hysteresis element is applied on the characterising equation, the equation can be split into two equations

$$\dot{x} = \quad 1 + u, \quad x \leq x_0, \tag{2.14a}$$
$$\dot{x} = -1 + u, \quad x \geq -x_0. \tag{2.14b}$$

The equations given in (2.14) leads to the description of a hybrid automaton. The location invariants are given by the discrimination of x compared to $\pm x_0$ as well as the guards are defined by these conditions. The graphical representation of the hybrid automata of the control system with hysteresis is sketched in Figure 2.11. The two location invariants are defined as an label below the corresponding states. In the defined hybrid automaton, only internally induced switchings are involved but no jumps.

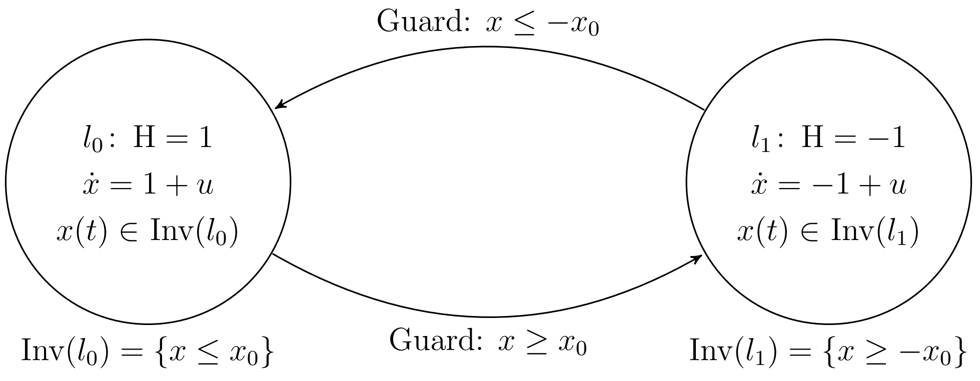

Guard: $x \leq -x_0$
l_0: H = 1
$\dot{x} = 1 + u$
$x(t) \in \mathrm{Inv}(l_0)$
l_1: H = −1
$\dot{x} = -1 + u$
$x(t) \in \mathrm{Inv}(l_1)$
$\mathrm{Inv}(l_0) = \{x \leq x_0\}$
Guard: $x \geq x_0$
$\mathrm{Inv}(l_1) = \{x \geq -x_0\}$

Figure 2.11: Hybrid Automaton of the Control System with a Hysteresis Element included.

2.3.2 Bouncing Ball

One of the most popular examples in the field of hybrid modelling or state event modelling is the bouncing ball. Several books and publications start with this example, out of the field of physical modelling, due to the simpleness. In the basic setup and in the sense of introduction this is a suitable example to point out characteristics of hybrid systems. In this subsection a similar specification of the bouncing ball is assumed like in [38] or in [50] but with the simplification of a bouncing in a perfect vertically direction, as it is introduced in [5].

Assume a punctiform ball with a certain mass m which is ideally bouncing in a vertical dimension. The space which can be measured in this direction is denoted by $x\colon \mathbb{R}_0^+ \to \mathbb{R}_0^+,\ x = x(t)$. For the equation of motion the velocity $v\colon \mathbb{R}_0^+ \to \mathbb{R},\ v = v(t)$ is required. The velocity v and space x of a punctiform mass is related by $v = \dot{x}$. Furthermore, some basic principles from physics has to be applied to find the final equation of motion. The acceleration a acting on the mass is negative operating to the acceleration of gravity g and the acceleration is the derivative of the velocity $a = \dot{v} = \ddot{x}$. To form a state space description, the equations have to be converted via $x_1 = x$ and $x_2 = v$ to

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -g \end{pmatrix}. \tag{2.15}$$

For the state vector $\boldsymbol{x} = (x_1, x_2)^T$ the state space description turns out to be a simple linear one according

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\boldsymbol{x}(t) + \begin{pmatrix} 0 \\ -g \end{pmatrix}.$$

The description can be interpreted as an constant input vector $\boldsymbol{u}(t) = g$, which is acting only on the second component of the system.

The next step in the modelling process is the definition of the bounce. In the defined environment, the ball bounces when in the falling down phase the ground is touched, expressed by

$$x_1 = 0 \qquad \text{and} \qquad x_2 \leq 0.$$

When the bounce occur, the bouncing process loses a part of its energy. If t^* denotes the event time and $c \in (0, 1)$ the equation to model the energy loss is given by

$$x_2(t_+^*) := -c x_2(t_-^*),$$

as inspired by [5]. The equation expresses the allocation of a new value of one of the state variables and indices at the event time t^* the one–sided limits. In the formalism of the hybrid automaton, the notation has to be different. A possible description is the mapping

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ -cx_2 \end{pmatrix}.$$

As for the discussed hybrid automaton for a bouncing ball, this model change has to be included in the transition as illustrated in Figure 2.12.

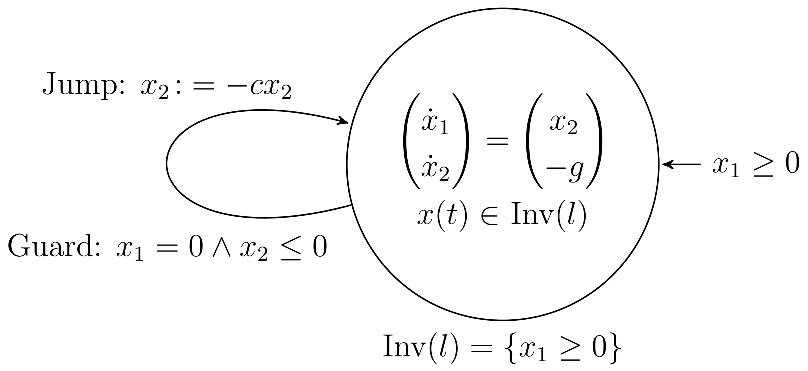

Jump: $x_2 := -cx_2$
$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -g \end{pmatrix}$
$x(t) \in \mathrm{Inv}(l)$
$x_1 \geq 0$
Guard: $x_1 = 0 \wedge x_2 \leq 0$
$\mathrm{Inv}(l) = \{x_1 \geq 0\}$

Figure 2.12: Hybrid Automaton of the Bouncing Ball Example.

As a final annotation for this example, the so called *Zeno*[1] *behaviour* of a bouncing ball, has to be addressed. Hybrid systems can take an infinite number of discrete transitions in finite time.

This behaviour is fulfilled by the bouncing ball as following characterised. Assuming an initial position $x(0) = x_0 > 0$ and an initial velocity $v(0) = v_0 = 0$ the analytic solution x of the differential equation $\ddot{x} = -g$ corresponding to the first order system can be computed, as published in [35]. According to the initial conditions the analytical solution is given by

$$x(t) = -\tfrac{g}{2}t^2 + v_0 t + x_0.$$

Due to the knowledge of the analytical solution the (first) zero of x as well as the velocity in this moment can be specified as

$$t_1 = \sqrt{\frac{2x_0}{g}} \qquad \text{and} \qquad \dot{x}(t_1) = -\sqrt{2gx_0}.$$

Because of the event condition $\dot{x}(t_+^*) := -c\dot{x}(t_-^*)$ and the analytically known structure of x it follows for the second falling phase the equation

$$x(t - t_1) = -\tfrac{g}{2}(t - t_1)^2 + c\sqrt{2gx_0}t + x(t_1),$$

where $x(t_1) = 0$. To locate the next zero the equation

$$-\tfrac{g}{2}(t - t_1)^2 + c\sqrt{2gx_0}t = 0$$

[1]The name is given according to Zeno's paradoxes, Zeno of Elea (ca. 490–430 B.C.). These paradoxes are a set of philosophical problems in general, like paradoxes of motion as Achilles and the tortoise.

has to be solved, which leads to

$$(t - t_1) = \tfrac{2}{g} c\sqrt{2gx_0} = 2c\sqrt{\tfrac{2x_0}{g}} = 2c\, t_1.$$

Thus the second zero in this bouncing process is located at

$$t_2 = (2c + 1)\, t_1.$$

Complete induction leads to the zero of the n–th bounce

$$t_n = \sqrt{\frac{2x_0}{g}} \left(-1 + 2\sum_{k=0}^{n-1} c^k \right) = \sqrt{\frac{2x_0}{g}}\, \frac{2c^n - c - 1}{c - 1}.$$

Now the limit $n \to \infty$ has to be computed to determine the time for an infinite number of bounces, as

$$t_\infty = \sqrt{\frac{2x_0}{g}}\, \frac{1 + c}{1 - c}.$$

This shows the Zeno behaviour of the bouncing ball, that infinite bounces are performed in a finite time t_∞.

2.3.3 Switched Circuit

The third example for hybrid systems is dealing with an electrical circuit where an ideal switching element is involved. The schematic is illustrated in Figure 2.13. The included switch separates two operating states from each other. If the switch is open and no connection is established, the circuit works similar to a serial oscillating circuit. If the switch is closed, the resistor and the capacity are short–circuited.

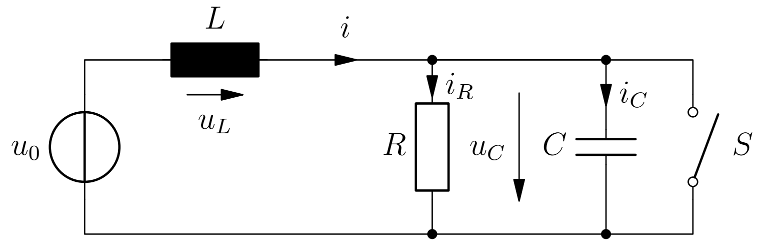


Figure 2.13: Example of an Electrical Circuit which includes an Ideal Switching Element.

The switch has two operating ranges: Either the switch is open which corresponds to the state $S = 0$ (`off`), or the switch is closed which corresponds to the state $S = 1$ (`on`).

For a opened switch $S = 0$, the equations which are involved to compute the mathematical model are given by the Ohm's law, the component equations

$$u_L(t) = L\,\frac{\mathrm{d}i}{\mathrm{d}t} \qquad \text{and} \qquad i_C(t) = C\,\frac{\mathrm{d}u_C}{\mathrm{d}t}$$

and the rules of Kirchhoff as

$$i(t) = i_C(t) + i_R(t)$$

and

$$u_L(t) + u_C(t) = u_0(t).$$

This results in the two formulas

$$\frac{\mathrm{d}}{\mathrm{d}t} u_C(t) = \frac{1}{C}\, i(t) - \frac{1}{RC} u_C(t), \tag{2.16a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} i(t) = -\frac{1}{L}\, u_C(t) + \frac{1}{L}\, u_0(t). \tag{2.16b}$$

Assuming the state vector $\boldsymbol{x} = (u_C, i)^T$ these equations can be reformulated to a linear state space description in the form of

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{pmatrix} \boldsymbol{x}(t) + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t). \tag{2.17}$$

Assuming now the switch $S = 1$ to be closed, the resistor R and the capacity C are short–circuited, as addressed in the beginning of this subsection. This intervention in the topology of the circuit results in a change in the model structure, cf. Figure 2.13 and assume the switch closed or better replaced by a short–circuit.

In the first mesh the voltage balance equation

$$u_0(t) - u_L(t) = 0$$

is valid, in the second mesh the short-circuit results in

$$u_C(t) = 0.$$

According to the component equations of the inductor it holds

$$\frac{\mathrm{d}}{\mathrm{d}t} i(t) = \frac{1}{L} u_0(t).$$

Both equation, together, form the model description for the same state vector as defined above for the second state $S = 1$ according to

$$\boldsymbol{x}(t) = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t). \tag{2.18}$$

To formulate the hybrid automaton for the switched circuit, one version of the model description is the form

$$\dot{\boldsymbol{x}}(t) = A_S \boldsymbol{x}(t) + B\boldsymbol{u}(t),$$

with matrices $A_S \in \mathbb{R}^{2\times 2}$ and $B \in \mathbb{R}^{2\times 1}$ given for the two states $S = 0, 1$ in equation (2.17) and in (2.18) the matrix A_S is the zero matrix. The parameter vector is given by $\boldsymbol{p} = (R, L, C)^T$, whereas algebraic variables are not present in this example.

The assumed model description from the point of a hybrid system is wrong. The input vector in this example starts to be interesting and as well the variable $S \in \{0,1\}$ indicates the state of the switch and is part of the input vector, as the input voltage u_0. Both have to be choosen via a user defined input. As a consequence, the input vector is defined as $\boldsymbol{u} = (u_0, S)^T$ and the formulation of the model structure is given via

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} = \begin{pmatrix} \frac{1}{RC}(S-1) & \frac{1}{C}(1-S) \\ \frac{1}{L}(S-1) & 0 \end{pmatrix}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t). \tag{2.19}$$

Furthermore, for the hybrid automata, the location invariants have to be defined. For this the two active states are defined by two states of the switch and these are defined by the input. Here occurs the problem of the formulation for the hybrid automata, the location invariants are dependend on the input vector. This is not covered in the definition of the hybrid automata, therefor a generalisation is needed to cover as well model of this and similar types.

To give an impression of the structure of the hybrid model (2.18) for the switched circuit the automaton using

$$A_S = \begin{pmatrix} \frac{1}{RC}(S-1) & \frac{1}{C}(1-S) \\ \frac{1}{L}(S-1) & 0 \end{pmatrix}, \qquad S \in [0,1] \qquad \text{and} \qquad B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

is illustrated in Figure 2.14.

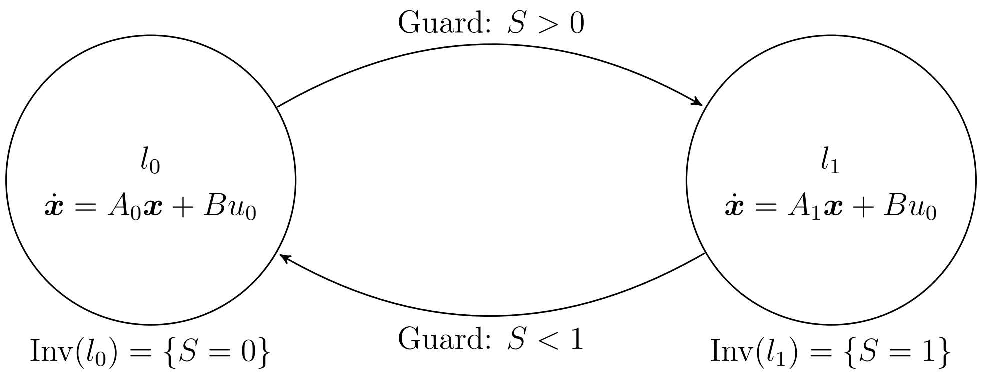

Guard: $S > 0$
l_0
$\dot{\boldsymbol{x}} = A_0\boldsymbol{x} + Bu_0$
l_1
$\dot{\boldsymbol{x}} = A_1\boldsymbol{x} + Bu_0$
$\mathrm{Inv}(l_0) = \{S = 0\}$
Guard: $S < 1$
$\mathrm{Inv}(l_1) = \{S = 1\}$

Figure 2.14: Hybrid Automaton of the switched circuit example.

2.3.4 Discussion and Remarks

The presented examples illustrate that the system theory of state event modelling and the theory of hybrid automata are closely related. Some aspects of the modelling process addresses the system formulation, some the framework of automata. State event modelling partly addresses as well the numerical calculations which are related to the simulation environment and the simulation run.
To sum up, the items of finding the valid location invariant and defining Jumps and Guards are even in academic examples challenging. Furthermore, performing the event action in case of changing some model aspects is close related to the modelling approach.

Several aspects in the example of the switched circuit addressed more analysis. The switch S represented by the values $\{0, 1\}$ describes the state of the switch and was part of the input vector. This resulted in the fact that the location invariants are depending on the input variable (external cvariable), which is not covered in the basic definition of a hybrid automaton. Furthermore, a more flexible structure in the environment regarding model description and the corresponding state space would be useful. The next chapter will provide a profound analysis of the hybrid structure of a model and provide a more general definition of hybrid automata for different purposes.

The introduced environment to define a hybrid model via theory of automata combined with system theory is intuitive, matches to first principal modelling and is a proper method to do analysis of certain model structures.

Chapter 3

Classification and Generalisation of Dynamic Hybrid Systems

In section 2.3 a couple of examples of hybrid dynamic systems are introduced. The hybrid behaviour can be generated by several attributes of the system. There are possibilities to provoke a transition via the input of a certain system, an internal condition can be fulfilled and the guard leads to a transition or a hard restriction triggers a jump. It is obvious that there are several reasons how to enable a transition in a hybrid dynamic system.

In literature and publications are suggested several specifications for certain subsets of hybrid dynamic systems, which characterise this particular behaviour.
This chapter is dedicated to characterisations for different type of behaviour in the hybrid dynamic systems, guided by known specifications applied to the classifications in the present mathematical environment.
This classification will help later on in the considerations to define a characterisation of state events and their state event actions in the environment of state event models for hybrid dynamic system.

In this chapter the hybrid automata of Definition 2.10 is a reference but the definition has to be modified according to system theory attributes. For the formulation of controlled switchings and jumps the input vector $\boldsymbol{u}$ has to be explicit present to provide a understandable formulation.

Definition 3.1. Denote $\mathcal{F}$ as the set of differential and algebraic equations, i.e. the set where the elements are tuples $(f, g) \in \mathcal{F}$ and denote functions as $f\colon\ \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R}$ the right side of the differential equation

$$\dot{x} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t)$$

and $g\colon\ \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^u$ represents the algebraic equation

$$g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}.$$

Equipped with this definition, Definition 2.10 will be reformulated according to continuous system theory. To distinguish the abstract hybrid automaton and the reformulated version, the following version for a dynamic hybrid system will be defined.

Definition 3.2 (Dynamic Hybrid System Automaton, DHSA)**.** Assume Definition 2.10 but consider the following changes:

(a) $X \subseteq \mathbb{R}^n$ is the *continuous state space* of the hybrid automaton in which the *continuous state vector* $\boldsymbol{x}$ takes their values.

(b) Act: $L \to \mathcal{F}_L,\ l \mapsto (f_l, g_l)$ is a mapping that assigns to each location $l \in L$ a set of differential and algebraic equations $(f_l, g_l) \in \mathcal{F}_L$, relating the state vector $\boldsymbol{x}$, the derivative of the state vector $\dot{\boldsymbol{x}}$, the algebraic vector $\boldsymbol{z}$ and the parameter vector $\boldsymbol{p}$ via

$$\begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}. \end{cases} \tag{3.1}$$

The solutions of these (explicit) differential and algebraic equations are called the *activities* of the location l.

The tuple $(L, X, A, W, E, \mathrm{Inv}, \mathrm{Act})$ with, here specified, Act mapping is called a *dynamic hybrid system automaton*.

Note 3.3. The definition of DHSA is used to point out the system theory approach. Of course some generality is loosing but common simulation environments support only this explicit specification. Additionally, in the system theoretical point of view as well the output equation $\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t)$ is missing. This will be added in the section where it will be used for some system characterisation.

The following sections will discuss the controlled and autonomous versions of switchings and jumps. Furthermore structure–variant systems will be considered more in detail due to their importance in the field of hybrid dynamic systems. The aim of this chapter will be a structural analysis of different architectures of general hybrid systems.

3.1 Switching and Jump Phenomena

The term switching relates to a transition in a hybrid model with the change in a certain dynamics of the model description, whereas the term jump means a direct influence of the state (trajectory) $\boldsymbol{x}$. The attribute autonomous, specifies that the change is induced internally and in contrast controlled, means that some external commands influenced the change. An overview about the characterisation is given in Table 3.1.

3.1.1 Autonomous and Controlled Switching

Switching phenomena are divided in autonomous and controlled. A similar description can be externally induced or internally induced. The internally induced switching process is nothing else than the event leaded by executing guards and location invariants. Different therefrom an externally induced switching is done by a control command.

	SWITCHING	JUMPS
AUTONOMOUS	Internally induced change of the dynamics	Internally induced discontinuously change of the state
CONTROLLED	Externally induced abruptly change of the dynamics	Externally induced discontinuously change of the state
	external control commands	

Table 3.1: First Characterisation of Switchings and Jumps in Hybrid Systems.

Autonomous Switching. The following formulation will describe the phenomena from a system theoretic point of view, motivated by aspects from [13]. Assume a certain state of the model is given, as in Definition 3.2. Assuming a certain location l and the corresponding model dynamic are given through

$$\begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}. \end{cases}$$

The transitions from a location l to a location l' are determined via the guards and the location invariants. Consider an arbitrary path inside the hybrid automata starting from the location l_1. This path can be characterised by the *sequence of locations*, denoted as

$$\boldsymbol{\ell} = \{l_1, l_2, \ldots, l_{N+1}\}, \qquad l_k \in L. \tag{3.2}$$

This sequence of locations $\boldsymbol{\ell}$ implies a so-called *switching time sequence*

$$\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_N\}, \tag{3.3}$$

the vector of even times of transitions between two locations l_k and l_{k+1} for $k = 1, \ldots, N$. The sequence of locations together with the switching time sequence is called the *switching schedule*

$$\mathcal{S} = (\boldsymbol{\ell}, \boldsymbol{\tau}). \tag{3.4}$$

The term autonomous indicates the internal induced switching from one state to another by no external influences. An illustration is given in Figure 3.1.

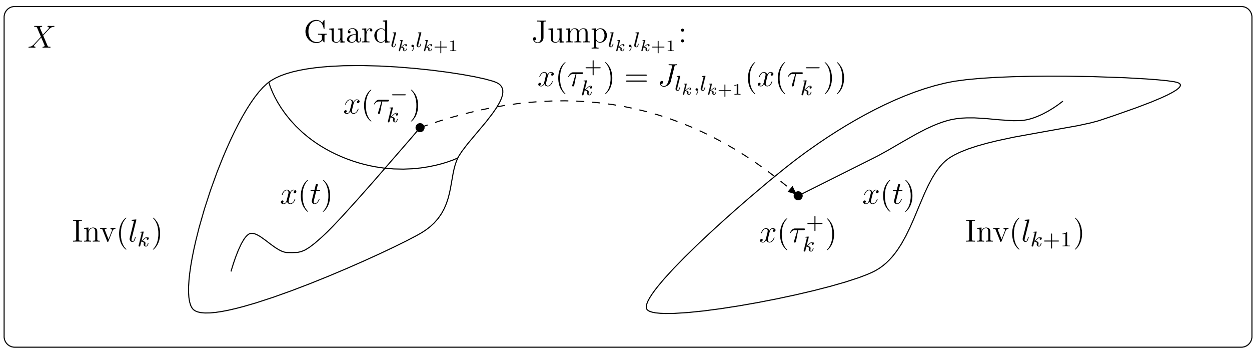

X
$\mathrm{Guard}_{l_k,l_{k+1}}$
$\mathrm{Jump}_{l_k,l_{k+1}}$:
$x(\tau_k^+) = J_{l_k,l_{k+1}}(x(\tau_k^-))$
$x(\tau_k^-)$
$x(t)$
$\mathrm{Inv}(l_k)$
$x(t)$
$x(\tau_k^+)$
$\mathrm{Inv}(l_{k+1})$

Figure 3.1: Autonomous Switching Process between Location l_k and l_{k+1} Illustrated by each of the Location Invariants, including one of the Guard Region in X.

A closer look to the illustration in Figure 3.1 shows, the value of the state before the event, at time τ_k, and after the event are in general different. This difference is covered by the description of the jump associated to the transition from l_k to l_{k+1}. For $\boldsymbol{x} \in X$ the state jump specification can be specified via a *state jump function* $J_{l_k,l_{k+1}} \colon X \to X$ as

$$x(\tau_k^+) = J\left(x(\tau_k^-)\right). \tag{3.5}$$

Remark 3.4. Some additional comments regarding the considered autonomous switching:

(a) The transition is not necessary oriented between two neighbour locations l_k and l_{k+1}. In general the transition can take place between two arbitrary locations l_k and l_j. The presented framework is oriented to the illustration in Figure 3.1 and hence neighbour locations are considered.

(b) Consequently the state jump function in general is defined between two arbitrary locations l_k and l_j as $J_{l_k,l_j} \colon X \to X$.

(c) The state jump function is a restriction to the jump condition. In Definition 3.2 the defined relation $\mathrm{Jump}_{l,l'}$ is a more generalised approach. For the classification in this chapter the state jump function is easier to handle. In the following chapters and subsection it will be noted if the jump relation or the jump function is used.

(d) The redefinition of a state jump by using the state jump function $J_{l_k,l_{k+1}} \colon X \to X$ is especially used in control problems. For an optimal formulation of the control problem, the functional representation is more suitable, cf. [52].

Autonomous jumps can be understood as the internally dynamics of a running dynamic hybrid system where no external influence is effecting.

Controlled Switching. As noted in Table 3.1, for controlled switching, the dynamics changes abruptly according to external control commands. According to the defined environment above, for the autonomous case, the switching in the controlled setup is defined externally via the *sequence of locations.* This sequence is not given by the guards and location invariants as before, it is given via an external controlled command. The *external defined sequence of locations* will be as well denoted by $\ell = \{l_1, l_2, \dots, l_{N+1}\}$, $l_k \in L$ in case of a predefined length of control commands. Concerning real time control, the sequence can be as well assumed to be infinite, if the system is perpetual active always controlled. The sequence of locations can be represented in this case by $\ell = \{l_1, l_2, \dots, l_k, \dots\}$. Each transition in the sequence of locations is enabled by a *sequence of control commands*

$$\boldsymbol{c} = \{c_1, c_2, \dots, c_N\}. \tag{3.6}$$

These control commands abruptly change the location of the system. In the simplest variant, the sequence of control commands can be interpreted as the discrete symbol which indicates the next location. In this case, as well $c_k \in L$ is valid.

The sequence of locations, respectively the sequence of control commands $\boldsymbol{c}$, implies directly the switching time sequence

$$\boldsymbol{\tau} = \{\tau_1, \tau_2, \dots, \tau_N\}.$$

The sequence of control commands together with the switching time sequence is called the *controlled switching schedule*

$$\mathcal{S}_c = (\boldsymbol{c}, \boldsymbol{\tau}). \tag{3.7}$$

A controlled switching between two arbitrary locations l_k and l_{k+1} in a certain sequence of control commands and locations is illustrated in Figure 3.2.

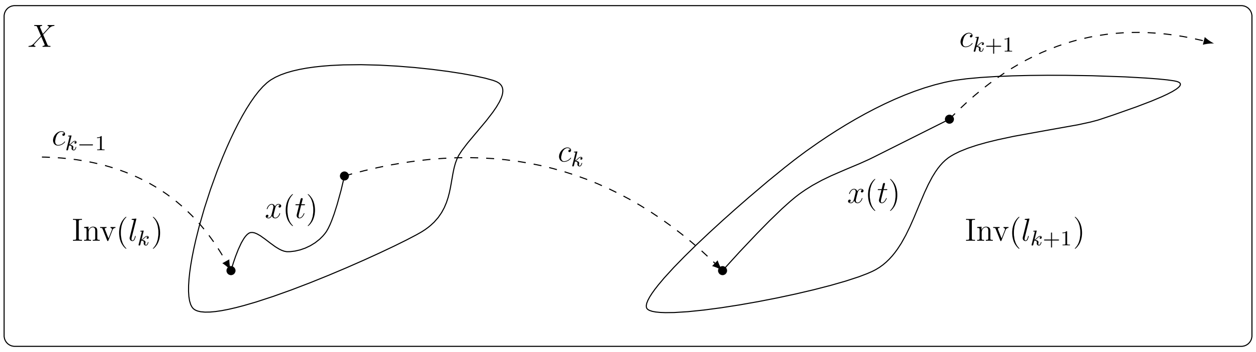


Figure 3.2: Controlled Switching Process between Location l_k and l_j Illustrated by each of the Location Invariants in X and the Effect of the external Control Commands.

The controlled switching is as well called *time–dependent switching*, respectively *timed switching systems* or *timed automata*, see [36]. This outlines the fact that the time of switching from one location to another location is defined by the external control and not by an internal restriction regarding the dynamics. The controlled switching is simpler to handle in numerical questions, because the event time is defined and does not have to be identified by numerical algorithms.

Of course there are combinations of autonomous and controlled switching possible. The matching notation for both switching situations is helpful, but the environment of an DHSA does not cover all specified requirements. For this purpose the DHSA has to be generalised, which will be done as a final step in this chapter.

3.1.2 Autonomous and Controlled Jumps

The term jumps, in context of dynamic systems, is always dedicated to a discontinuous change of the state vector $\boldsymbol{x}$. In principle this behaviour is covered by the Jump_{l_j,l_k} relation of an hybrid automaton together with the region Guard_{l_j,l_k} where the jump is to be executed. In this subsection, the jump will be analysed by itself and this means the particular dynamics behind is not considered. For system simulation, a jump will be defined and investigated by themselves as an autonomous versus a controlled execution.

Autonomous Jumps. For considering autonomous jumps, first the so-called *autonomous jump set* $\mathcal{S} \subset X$ has to be specified. $\mathcal{S} \subset X$ is a region where the jump of the state is executed, see Figure 3.3. Assume $\tau \in \mathbb{R}$ to be the moment where

$$\boldsymbol{x}(\tau) \in \mathcal{S}$$

is achieved, the time value τ is the *event time* of the jump. For the execution of the state jump again a *state jump function* $J\colon X \to X$ is responsible. The state jump execution at the event time τ is written as

$$\boldsymbol{x}(\tau^+) = J\left(\boldsymbol{x}(\tau^-)\right). \tag{3.8}$$

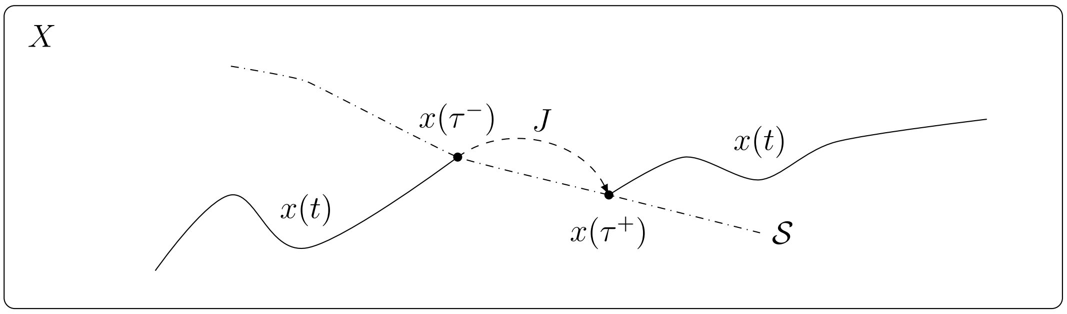


Figure 3.3: Autonomous Jump Process at a certain Jump Set $\mathcal{S}$ illustrated in X.

Remark 3.5. Some remarks according to autonomous jumps:

(a) The mapping $J\colon X \to X$ represents in principal the same state jump as in the case of switchings, but the labeling of the locations is lost. Due to the consideration of jumps by itself, the connection to the locations is suppressed in this subsection.

(b) The addressed event time is closely related to the event definition of subsection 2.2.2 State Event Modelling. In chapter 4, State Event Handling in the Numerical Environment, this aspect will be picked up again.

Assume more than one autonomous jump set, a family of sets $\{S_i\}$ for $i = 1, \dots, k$ the state space is partitioned in several divisions. Consider a certain dynamics of the state vector in this divided subspace several state jumps will be performed, each of them at a certain time instant τ_i. This can be summarised in a similar way as it was done for switching processes. Define a sequence

$$\boldsymbol{\tau} = \{\tau_1, \tau_2, \dots, \tau_N\}$$

of all time instants where a state jump is executed. $\boldsymbol{\tau}$ is called the *jumping time sequence*.

This jumping time sequence is the same like in the case of switching and will be the key point of interest in the issue of state event finding. The environment in the case of several autonomous jump sets is matching to the definition of a DHSA. The resulting divisions of the state space can be associated with the guard regions and the location invariants. In principle, the autonomous jumps could occur as well inside a location invariant this would be the case if a jump is executed but the dynamic is not changed. For this purpose, a transition to the location by itself can be used in the environment of a DHSA.

Controlled Jumps. Analog to controlled switching as well analog jumps are depending on external control commands. These control commands result in a discontinuity in the state trajectory. To formalise this jumps, a *sequence of control commands* will be assumed as

$$\boldsymbol{c} = \{c_1, c_2, \dots, c_N\}.$$

Each command c_k of the sequence $\{\boldsymbol{c}\}$ has to be selected in a *control alphabet*

$$\mathcal{C} = \{c_1, c_2, \dots, c_K\}.$$

A command $c_k \in \mathcal{C}$ enables a state jump which is formalised via the *controlled state jump function* $J_{c_k}\colon X \to X$. Due to the setup, the state jump functions must be already defined, depending on external commands and for each command in this alphabet a corresponding state jump function will be executed, illustrated by an example in Figure 3.4.

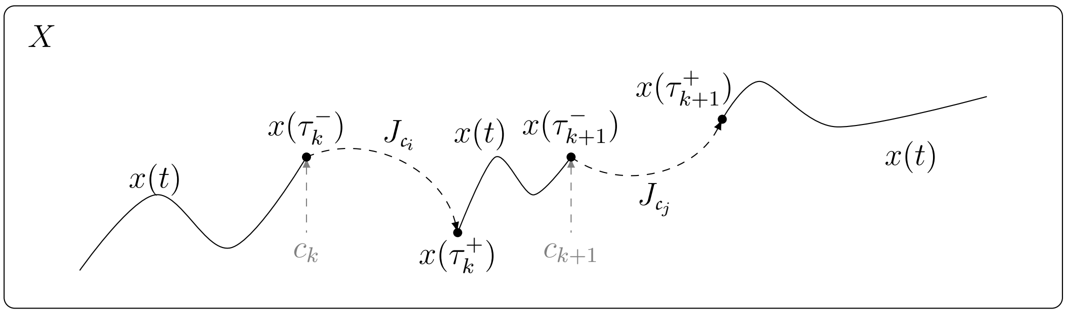


Figure 3.4: Controlled Jump Processes enabled via Control Commands $c_k = c_i$ and $c_{k+1} = c_j$ illustrated in X.

The sequence of control commands $\boldsymbol{c}$ implies directly the *jumping time sequence*

$$\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_N\}.$$

In the same way as it was done in the controlled switching process, the sequence of control commands and the jumping time sequence define a schedule for the controlled jumps.

3.1.3 Discourse regarding Mathematical Framework

Chapter 2, Mathematical Framework for Dynamic Hybrid Models, was dedicated to forming the mathematical environment for the first attempt of a description for dynamic hybrid models in the field of state event modelling. The final section shows several examples where the environment is suitable, but in which as well a problem occurs. The switched circuit shows a weak point in the case of location invariants which are depending on the input of the system.

The third chapter Classification and Generalisation of Dynamic Hybrid Systems started a classification of hybrid phenomena and evaluats the so far defined mathematical framework, regarding the potential of description of this phenomena. The main classification is given regarding switching of the dynamics and state jumps. Here common literature suggest in both cases autonomous and controlled version of these processes. The previous subsections presents the mathematical setting of this phenomena and the embedding of the phenomena in the mathematical framework of a dynamic hybrid system automaton. Basically, the autonomous versions of the both processes fulfill the constrains of the framework but the controlled versions do not. For the controlled versions, an external control command, respectively a sequence of external control $\{\boldsymbol{c}\}$ commands are necessary and are not covered in the definition of DHSA.

To summarise the first studies, the mathematical environment has problems with incidence of manipulating input variables. Especially in control applications, these external control inputs are not possible to proceed without this input. This is a first requirement in the generalisation in the concept of DHSA.

3.2 Structure–Variant Systems

Before entering in the aspects of system architecture and generalisation of the given approaches, an important excursus in the field of *structure–variant* or *structure–variable systems* takes place. This term is not related to the dynamic hybrid system theory and it is out of the field of system theory and control engineering. The definition will as well be given in an system theoretic approach. This section is dedicated to classify and rank this systems in terms of dynamic hybrid systems.

These kind of systems are presented in the literature quite early, starting from 1950 in the Soviet Union. One example is [14], where the relationship between variable structure systems and the zeros of a linear time–invariant multivariable system, similar defined as in equation (2.3), are investigated. The particular variable structure is defined as a discontinuous control action which changes the structure upon reaching a set of switching surfaces. The discontinuous control action is performed via a discontinuous input vector $\boldsymbol{u}$ at the moment of reaching the switching hyperplanes. Related investigations are raised decades later, in 2002, in [39], and the topic is still relevant in control theory as shown in [46].

In this section, a brief introduction about structure–variant systems will be given and the connection to dynamic hybrid systems will be explained. The starting point is a simple example out of control theory.

Example 3.6. Consider, as in [50], a control system described by the equation

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u), \tag{3.9}$$

where $u\colon \mathbb{R} \to \mathbb{R}$ is the scalar control input and $\boldsymbol{x}\colon \mathbb{R} \to \mathbb{R}^n$ the state of the system. Assume a state feedback control law is applied such as

$$u(t) = \begin{cases} \varphi_1(\boldsymbol{x}(t)), & \text{when } y > 0, \\ \varphi_2(\boldsymbol{x}(t)), & \text{when } y < 0, \end{cases}$$

for $\varphi_i\colon \mathbb{R}^n \to \mathbb{R}$ $(i = 1, 2)$ and the scalar output function $y(t) = h(\boldsymbol{x}(t))$, where $h\colon \mathbb{R}^n \to \mathbb{R}$. Denote

$$f_i(\boldsymbol{x}) := f(\boldsymbol{x}, \varphi_i(\boldsymbol{x}))$$

for $i = 1, 2$. The dynamical system results in

$$\dot{x} = \begin{cases} f_1(\boldsymbol{x}) & \text{if} \quad h(\boldsymbol{x}) < 0, \\ f_2(\boldsymbol{x}) & \text{if} \quad h(\boldsymbol{x}) > 0. \end{cases} \tag{3.10}$$

This system description is known as variable–structure system.

To define the automaton environment, this description has to be translated to find the DHSA. The system works in two different locations distinguished by the actings f_1 and f_2 in the dynamics and the expressions $h(\boldsymbol{x}) \geq 0$ and $h(\boldsymbol{x}) \leq 0$ serve as the location invariants of these two locations. One problem in this example is, the specification of the trajectories of the system. This shows that this example is closer to the state event modelling approach where also the solutions are computed numerically and are not available before.

Nevertheless some discussions about the solutions are useful. The combined vector field

$$f(\boldsymbol{x}) = \begin{cases} f_1(\boldsymbol{x}) & \text{for } h(x) > 0, \\ f_2(\boldsymbol{x}) & \text{for } h(x) < 0, \end{cases}$$

is in general discontinues on the switching surface $h(x) = 0$. This means that the vector field f violates Theorem 2.15 about existence and uniqueness of solutions. One possibility is the reformulation to the equivalent integral equation as it is proposed in the approach of successive approximation, cf. [27]. The integral form is given by

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t f(\boldsymbol{x}(s))\mathrm{d}s.$$

Solutions of this integral equation do not require to be differentiable and are called solutions of (3.10) in the sense of Carathéodory, more in [11]. For this purpose the value of f on the surface $\{\boldsymbol{x} \in \mathbb{R}^n \colon h(\boldsymbol{x}) = 0\}$ has to be defined, at least for the one-sided values of the solutions, to perform the state jump. The (not completely equipped) DHSA is illustrated in Figure 3.5.

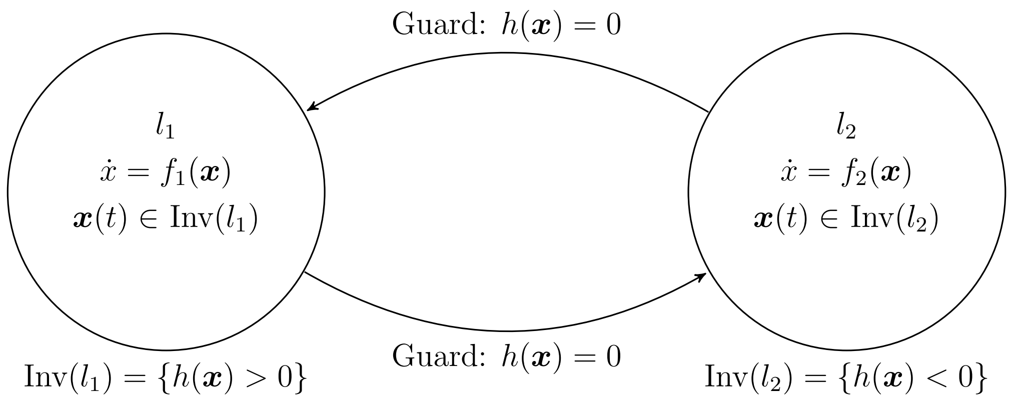

Guard: $h(\boldsymbol{x}) = 0$
l_1
$\dot{x} = f_1(\boldsymbol{x})$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_1)$
l_2
$\dot{x} = f_2(\boldsymbol{x})$
$\boldsymbol{x}(t) \in \mathrm{Inv}(l_2)$
Guard: $h(\boldsymbol{x}) = 0$
$\mathrm{Inv}(l_1) = \{h(\boldsymbol{x}) > 0\}$
$\mathrm{Inv}(l_2) = \{h(\boldsymbol{x}) < 0\}$

Figure 3.5: A two State Variable–Structure System illustrated by an Dynamic Hybrid System Automaton.

In principle the structure–variable systems are similar to the autonomous switching. The given example shows as well a possibility how to realise a controlled scenario.

The structure can be generalised from two, to several states as following; see [23]. Consider again the dynamics

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}),$$

with $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$. The vector field is assumed to be a piecewise continuous function $f \colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ in the sense

$$f(\boldsymbol{x}, \boldsymbol{u}) = f_i(\boldsymbol{x}, \boldsymbol{u}) \qquad \text{if} \qquad (\boldsymbol{x}, \boldsymbol{u}) \in \Omega_i, \qquad i = 1, \dots, N.$$

Thereby $\Omega_1, \dots, \Omega_N \subset \mathbb{R}^n \times \mathbb{R}^m$ are a closed partition of $\mathbb{R}^n \times \mathbb{R}^m$, this means for $i \neq j$

$$\Omega_i^\circ \cap \Omega_j^\circ = \emptyset \quad \text{and} \quad \bigcup_{i=1}^N \Omega_i = \mathbb{R}^n \times \mathbb{R}^m .$$

This generalisation to n states of the variable–structure system results in a DHSA with N locations. For the translation the boundary $\partial\Omega_i$ is necessary to formulate the location invariants and perform the state jumps.

3.3 Aspects of System Architecture

The objective of this section is the junction of the system simulation environment and the mathematical environment. The keyword – system architecture – points out the system properties regarding the surroundings and the location and interaction of the interior components. Mainly the focus is oriented in the separation between continuous and discrete modules in the mathematical framework.

The classification, up to now, gave an orientation which behaviour is covered by the mathematical framework and for which the framework needs a refinement. The architecture should support the right placement and the needed extension.

The first structure to be investigated is the hybrid automaton specified in Definition 2.10. In this environment, the continuous variables $\boldsymbol{x}$ and $\boldsymbol{w}$ are involved in the continuous dynamics of a location. Figure 3.6 illustrates the principle tasks of a hybrid system and connect it with the definitions of the mathematical environment.

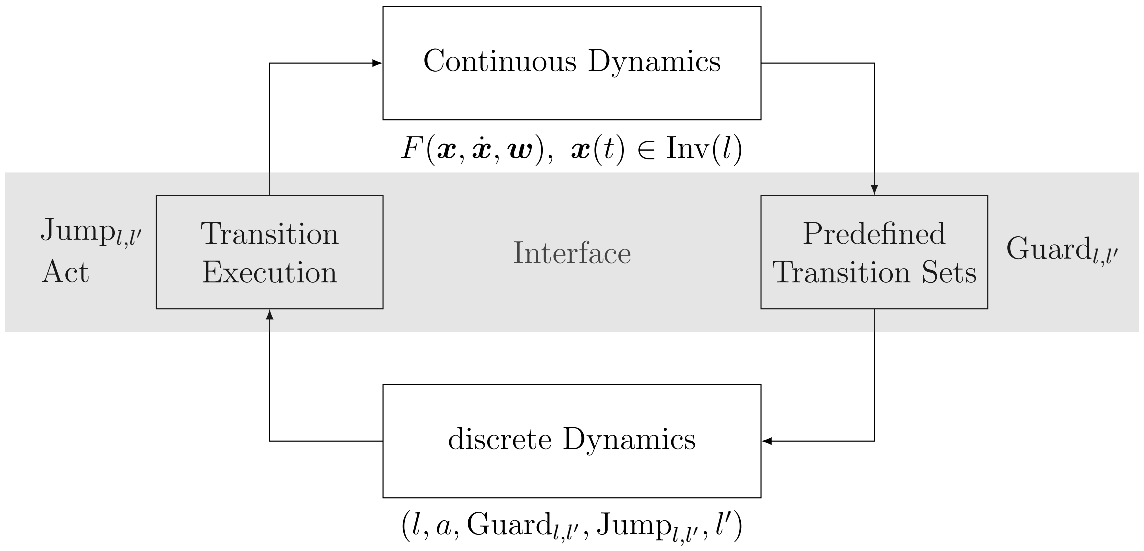

Continuous Dynamics
$F(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}),\ \boldsymbol{x}(t) \in \mathrm{Inv}(l)$
$\mathrm{Jump}_{l,l'}$ Act
Transition Execution
Interface
Predefined Transition Sets
$\mathrm{Guard}_{l,l'}$
discrete Dynamics
$(l, a, \mathrm{Guard}_{l,l'}, \mathrm{Jump}_{l,l'}, l')$

Figure 3.6: System Architecture labeled with the corresponding Mathematical Tools from Definition 2.10 of a Hybrid Automaton.

This first formulation of a principal architecture defines for the first time the term hybrid system, which will be given as following.

Definition 3.7 (Hybrid system)**.** A system architecture as illustrated in Figure 3.6 provided with the mathematical environment of a hybrid automaton in Definition 2.10 is called a *hybrid system*.

Following the strategy from the mathematical environment point of view, the next major step is the partition of the external continuous variables $\boldsymbol{w}$ into an input vector $\boldsymbol{u}$ and output vector $\boldsymbol{y}$, as written in Definition 2.7. Of course this refinement is not a major step because the influence of external continuous variables were given as well before. What is more important, is that the definition of an explicit input and output vector offers the possibility of external influences in therms of controlling the continuous dynamics. Furthermore, the continuous dynamics in a

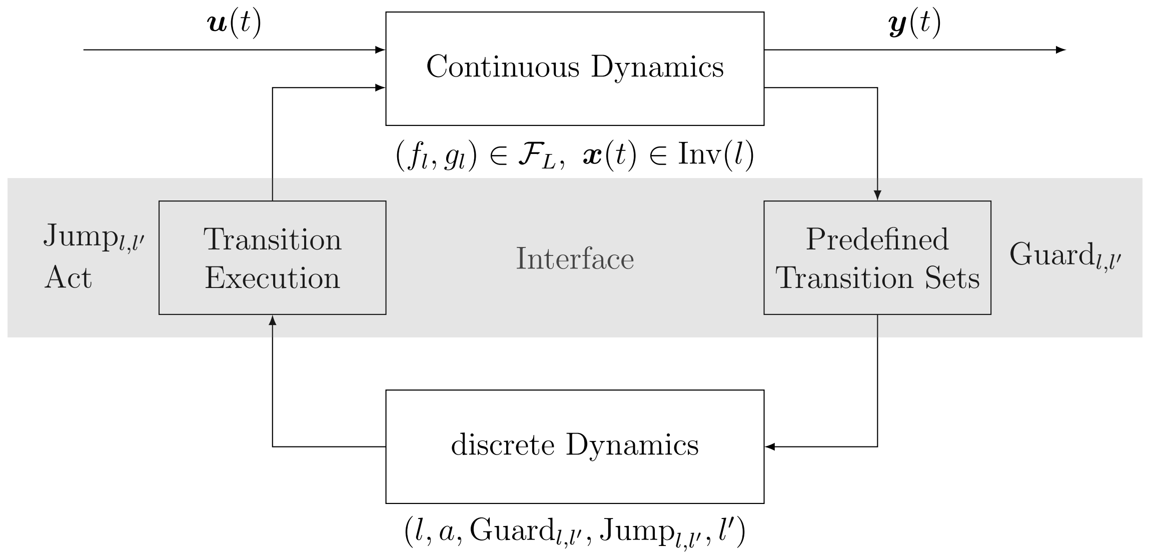


Figure 3.7: System Architecture of a Continuous Controlled Hybrid System labeled with the corresponding specified mathematical tools from Definition 3.2 of a Dynamic Hybrid System Automaton.

location l is formulated via an explicit system of first order ordinary differential equations and a system of algebraic equations (f_l, g_l). This new setup results in a refinement of the system architecture as illustrated in Figure 3.7. This structure allows to define a hybrid system which has control interfaces for the continuous dynamics.

Definition 3.8 (Continuous Controlled Hybrid System, CCHS)**.** A system architecture, as illustrated in Figure 3.7, provided with the mathematical environment of a DHSA – Definition 3.2 – is called a *continuous controlled hybrid system*.

Continuous controlled hybrid systems covers most of the discussed hybrid phenomena. The autonomous variants of switching and jumps are covered in this environment, as discussed in section 3.1, as well as the structure–variant systems introduced in section 3.2. As a resume up to now, the provided mathematical environment covers several system behaviour except the external controlled behaviour.

To include this behaviour as well, the system architecture has to offer the possibility to process as well discrete input variables. These discrete inputs can be further used to handle external control commands as used for the controlled switching as well for the controlled jumps. For this purpose the system architecture has to be refined to keep a continuous controlled system on the one hand and to achieve a discrete controlled system on the other hand together as a controlled hybrid system. The principle system architecture is illustrated in Figure 3.8.

The component which is missing to define a controlled hybrid system is the mathematical environment. As shown in section 3.1 the required mathematical equipment to formulate controlled jumps and switchings is neither given by a hybrid automaton nor by a dynamical hybrid system automaton. The next section will settle this enlargement of the mathematical framework regarding several aspects collected in the last sections and examples.

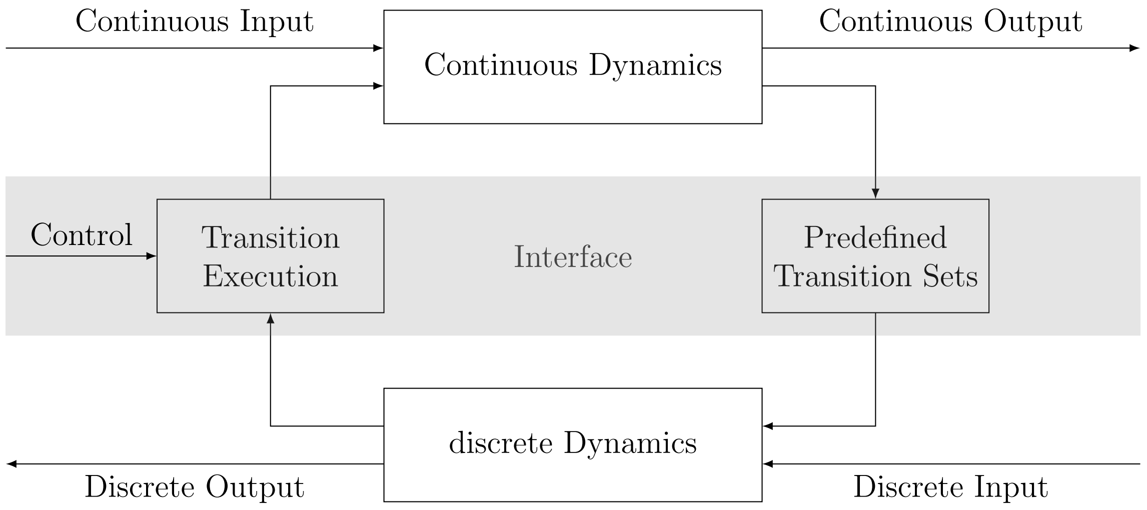


Figure 3.8: System Architecture of a Controlled Hybrid System labeling several Issues.

3.4 Generalisation of the Mathematical Framework

This section is dedicated to the completion of the mathematical framework. In the example of the switched circuit in subsection 2.3.3 it was shown, that in some cases the location invariants are depending on external input. This is not covered in the present definition of an hybrid automaton. Section 3.1 it is demonstrated that, the autonomous processes are covered by the framework of an DHSA, the controlled processes not. In the discussion about the architecture of a controlled hybrid system, this attribute is pointed out again.

This section is dedicated to close this gaps in the framework and formulate proper mathematical tools to use them for the characterisation of state event modelling.

3.4.1 Generalised Hybrid Automaton

The starting point for the first generalisation is Definition 2.10, the definition of the hybrid automaton. The following definition will refine the definition in order to balance the role of the continuous and discrete dynamics in a hybrid automata. Furthermore, the close related role of the location invariant and the guards, which are important for the discrete transitions will be connected to the edges of the automaton.

Definition 3.9 (Generalised Hybrid Automaton)**.** The definition of the set of discrete states L, the continuous state space X, the set of symbols to label the edges A, the continuous communication space W and mapping that assigns to each location $l \in L$ a set of differential algebraic equations $F_l \in F_L$ $\mathrm{Act}\colon L \to F_L$ are as given in Definition 2.10. Furthermore the set $R \subset (L \times X) \times (A \times W) \times (L \times X)$ is defined.
The sixtupel $(L, X, A, W, R, \mathrm{Act})$ is called a *generalised hybrid automaton.* As a consequence several concepts have to be redefined in the new setup.

A *continuous trajectory of a generalised hybrid automaton* (l, a, δ, x, w) associated with a location $l \in L$ and a discrete external symbol a includes a time of duration $\delta \in \mathbb{R}_0^+$ of the continuous trajectory, a piecewise continuous function $\boldsymbol{w}\colon [0,\delta] \to W$ and a continuous and piecewise differentiable function $\boldsymbol{x}\colon [0,\delta] \to X$ which fulfill

(a) $(l, \boldsymbol{x}(t), a, \boldsymbol{w}(t), l, \boldsymbol{x}(t)) \in R, \quad \forall t \in (0,\delta)$

(b) $F_l(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{w}(t)) = 0, \quad \forall t \in (0,\delta) \setminus \{\hat{t}_1, \ldots, \hat{t}_w\}$, where $\{\hat{t}_1, \ldots, \hat{t}_w\}$ is the set of discontinuity of $\boldsymbol{w}$ in $(0,\delta)$.

A *trajectory of the generalised hybrid automaton* is an (infinite) sequence of continuous trajectories

$$(l_0, a_0, \delta_0, w_0) \longrightarrow (l_1, a_1, \delta_1, w_1) \longrightarrow (l_2, a_2, \delta_2, w_2) \longrightarrow \ldots,$$

so that at the event times

$$t_0 = \delta_0, \quad t_1 = \delta_0 + \delta_1, \quad t_2 = \delta_0 + \delta_1 + \delta_2, \quad \ldots,$$

satisfy the relation

$$(l_j, \boldsymbol{x}_j(t), a_l, \boldsymbol{w}_j(t), l_{j+1}, \boldsymbol{x}_{j+1}(t)) \in R, \quad \forall j = 0, 1, 2, \ldots.$$

The subset R includes the location invariants, guards and jumps of the hybrid automaton in the following way:

(a) $\mathrm{Inv}(l) = \{(x, a, x) \in X \times A \times X\colon (l, x, a, w, l, x) \in \mathbb{R}\}$

(b) $\mathrm{Guard}_{l,l'} = \{(x, a, w) \in X \times A \times W\colon \exists x' \in X \text{ so that } (l, x, a, w, l', x') \in R\}$

(c) $\mathrm{Jump}_{l,l'}(x, a, w) = \{x' \in X\colon (l, x, a, w, l', x') \in R\}$

Furthermore define $H = L \times X$ the *hybrid state space* and $V = A \times W$ the set of *external hybrid variables* the set R fulfills $R \subset H \times V \times H$.

Remark 3.10. Remarks regarding the link to previous definitions:

(a) One important note is, the location invariant as well as the guard and jump set are all depending on the discrete and continuous external variables.

(b) The set of edges E is not longer present in the definition of the generalised hybrid automata. But it can be shown, that the set E can be recovered from a convenient set of R.

(c) The algebraic structure of $\mathrm{Jump}_{l,l'}$ is different to the previous definition. In the refined version it works more similar to the state jump function $J_{l,l'}$ defined in equation (3.5). It holds $\mathrm{Jump}_{l,l'}(X \times A \times W) \subset X$ as well as $J_{l,l'}(X) \subset X$.

(d) The interpretation of $R \subset H \times V \times H$ gives some more structural knowledge about the set R. If the external hybrid variables in V are not leading to the predefined transition set, the location will not be changed. If they guide in the predefined transition set a transition will be enabled.

This generalisation of a hybrid automaton guide to a wider range of applications. The generalised definition include a hybrid state, which consists of the state of the continuous dynamics as well as the state of the discrete dynamics as before, hence $(l, \boldsymbol{x})$. Furthermore, the external variables are now composed by continuous external variables $\boldsymbol{w} \in W$ and discrete external variable $a \in A$.

This generalisation of a hybrid automaton keep valid, if the continuous external variables will be split in input and output vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ as specified in Definition 2.7. Furthermore, the separation of the DAE function F_l in the explicit differential and algebraic equations (f_l, g_l) as introduced in Definition 3.2 does not change the given specification.

This generalisation provide the right framework to reconsider the example of a switched circuit as discussed in subsection 2.3.3. The problem which remains in the discussion of the hybrid automaton as a mathematical framework for this problem specification is the dependency on a input variable. Furthermore, the by nature discrete input is modeled as an continuous input to fit to the framework. With the generalisation of the hybrid automaton framework the possibility to solve this problem is given. The extension of the switched circuit from subsection 2.3.3 is given in the next example.

Example 3.11. Reconsider the switched circuit from Figure 2.13 and the model description derived in subsection 2.3.3 resulting in

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} = \begin{pmatrix} \frac{1}{RC}(S-1) & \frac{1}{C}(1-S) \\ \frac{1}{L}(S-1) & 0 \end{pmatrix} \begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t).$$

For the formulation of the model $\dot{\boldsymbol{x}} = A_S \boldsymbol{x} + B u_0$ in one of the two locations the matrices

$$A_S = \begin{pmatrix} \frac{1}{RC}(S-1) & \frac{1}{C}(1-S) \\ \frac{1}{L}(S-1) & 0 \end{pmatrix}, \qquad S \in \{0,1\} \qquad \text{and} \qquad B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

are used.

To customise this model description to the framework of a generalised hybrid automata the following sets are defined.

$$L = \{l_0, l_1\}, \quad X = \mathbb{R}^2, \quad A = \{0,1\},$$

W is equal $\mathbb{R}$ because the continuous external variable only consists of the input u_0 and Act is mapping l_i onto $\dot{\boldsymbol{x}} = A_i \boldsymbol{x} + B u_0$. Thus R is defined as in the environment demanded location invariants, guards and jumps can be defined in this example due to the small number of locations for each situation. The approach in this example is a reverse engineering. The location invariant can be described by

$$\begin{aligned} \operatorname{Inv}(l_0) &= \{(\boldsymbol{x}, 0, u_0) \in X \times \{0,1\} \times \mathbb{R} \colon (l_0, \boldsymbol{x}, 0, u_0, l_0, \boldsymbol{x}) \in R\} \qquad \text{and} \\ \operatorname{Inv}(l_1) &= \{(\boldsymbol{x}, 1, u_0) \in X \times \{0,1\} \times \mathbb{R} \colon (l_1, \boldsymbol{x}, 1, u_0, l_1, \boldsymbol{x}) \in R\}. \end{aligned}$$

Similar is the approach for the Guards of the locations

$$\begin{aligned} \mathrm{Guard}_{l_0,l_1} &= \{(\boldsymbol{x}, 1, u_0) \in X \times \{0,1\} \times \mathbb{R} \colon (l_0, \boldsymbol{x}, 1, u_0, l_1, \boldsymbol{x}) \in R)\} \qquad \text{and} \\ \mathrm{Guard}_{l_1,l_0} &= \{(\boldsymbol{x}, 0, u_0) \in X \times \{0,1\} \times \mathbb{R} \colon (l_1, \boldsymbol{x}, 0, u_0, l_0, \boldsymbol{x}) \in R)\}. \end{aligned}$$

In this case, the jumps are not considered, because there is no jump in this example present. This switched circuit system is a variable–structure system.

Remark 3.12. In the given model structure, there are only two states distinguished. A closer look to the model description in location l_1 shows, that the first equation of the differential equation system

$$\dot{u}_C(t) = 0$$

is a trivial case of a differential equation. In this case the vector space of the state vector changes the dimension, which is recognisable in the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t).$$

In the investigated mathematical environment this is currently not covered in the model description. It would be useful to add this feature, because it is not useful from a numerical point of view to simulate a state space model, where one or more components are equal to zero.

The concept of generalised hybrid automaton allows to define an interesting algebraic operation. It permits the definition of a composition of generalised hybrid automata.

Definition 3.13. Consider two generalised hybrid automata $H_1 = (L_1, X_1, A, W, R_1, \mathrm{Act}_1)$ and $H_2 = (L_2, X_2, A, W, R_2, \mathrm{Act}_2)$ with shared external hybrid variables. The *composition of two generalised hybrid automata* is given by

$$H = H_1 \circ H_2 = (L, X, A, W, R, \mathrm{Act}),$$

where

$$L = L_1 \times L_2, \qquad X = X_1 \times X_2, \qquad \mathrm{Act} = \mathrm{Act}_1 \times \mathrm{Act}_2$$

and for elements $((l_1, l_2), (\boldsymbol{x}_1, \boldsymbol{x}_2), a, w, (l_1', l_2'), (\boldsymbol{x}_1', \boldsymbol{x}_2')) \in (L \times X) \times (A \times W) \times (L \times X)$ it is

$$R = \{((l_1, l_2), (\boldsymbol{x}_1, \boldsymbol{x}_2), a, w, (l_1', l_2'), (\boldsymbol{x}_1', \boldsymbol{x}_2')) : (l_i, \boldsymbol{x}_i, a, w, l_i', \boldsymbol{x}_i') \in R_i,\ i = 1, 2\}.$$

Remark 3.14. The composition of two generalised hybrid automata has some limitations. Following some remarks according this aspects:

(a) The external hybrid variables must coincide, this means both automata has the same connection to external environment.

(b) The definition of the relation R follows the idea of a direct product in the definition.

(c) The composition is sometimes denoted as *synchronous parallel composition*, which indicate better the structural background of the composition, as it is done in [50].

(d) More topological and algebraic information about the composition of generalised hybrid automata is given in [25].

The generalisation presented in this subsection is one of two steps to provide the mathematical environment for the system architecture presented and discussed in this chapter. The last step is to catch the part of controlled hybrid systems, which will be done in the following subsection.

3.4.2 Generalised Dynamic Hybrid Systems

This last subsection is dedicated to the incorporation of the last missing aspects for a controlled hybrid system:

(a) The possibility of realising autonomous and controlled jumps and switchings.

(b) For each location different state spaces are available.

For this purpose a last redefinition of the mathematical environment will be introduced in two steps. First the formulation of a dynamic hybrid automata will be generalised regarding the connection of the state space to the location. This makes it possible to have different state spaces in different locations. The second step is the adding of the controlled environment of jumps and switchings. This generalisation is motivated by the work of [5] and oriented to provide the environment for the goal of this thesis, the state event characterisation.

Starting from the system theoretical point in one location $l \in l$, the continuous dynamics is given in first instance by the state space X, the continuous communication space W and the differential algebraic equation F_l. To allocate this dynamics to the state the notation will change to X_l the state space associated with the location l and the state space vector $\boldsymbol{x}_l \in X_l$ as well as the continuous communication space W_l with the corresponding continuous external variables $\boldsymbol{w}_l \in W$. To sum up, the dynamics in a certain location $l \in L$ is given by

$$(X_l, F_l),$$

where the continuous communication space W_l is preliminary not influencing, so $\boldsymbol{w}_l \in W_l$ is assumed to be constant. Starting from here the following definition can be done.

Definition 3.15 (Generalised Dynamic Hybrid System Automaton, GDHSA)**.** Let L be a finite set of locations $l_1, \ldots, l_\kappa$, X_l the state space associated to the location l and W_l the continuous communication space. Furthermore F_L is the set of functions $F_l \in F_L$, which links the state vector $\boldsymbol{x}_l \in X_l$ and the external continuous variables $\boldsymbol{w}_l \in W_l$ of a location l according to $F_l(\boldsymbol{x}_l, \dot{\boldsymbol{x}}_l, \boldsymbol{w}_l) = 0$.

(a) Denote $\boldsymbol{X} = \{X_l\}_{l \in L}$, $\boldsymbol{W} = \{W_l\}_{l \in L}$ the collection of the corresponding spaces.

(b) Define $S = \bigcup_{l \in L} (\{l\} \times X_l)$ as the *hybrid state space*, and $S_l = \{l\} \times X_l$ as the component according to the location l.

(c) The so far guard region will be characterised via a collection of *autonomous jump sets* $\boldsymbol{G} = \{G_l\}_{l \in L}$, where $G_l \subset X_l$.

(d) Transitions, up to now covered by the jump relation, are specified by the collection of *autonomous jump transition maps* $\boldsymbol{J} = \{J_l\}_{l \in L}$, with mappings $J_l \colon G_l \to S$.

The sixtupel $(L, \boldsymbol{X}, \boldsymbol{W}, F_L, \boldsymbol{G}, \boldsymbol{J})$ is called a *generalised dynamic hybrid system automaton $\mathcal{H}$.*

Following from this definitions the topological structure of the given sets can be analysed further. The union of all sets $G_l \times \{l\}$ is called *the autonomous jump set*

$$G = \bigcup_{l \in L} (\{l\} \times G_l)$$

and the mapping $J\colon G \to S$ is called *the autonomous jump transition map.* Moreover the destinations of the jumps are interesting. These are characterised by the collection of *jump destination sets* $\boldsymbol{D} = \{D_l\}_{l\in L}$, where[1]

$$D_l = \pi_1 \left(J(G) \cap S_l\right).$$

The last interest is to formalise the answer to the question, which transitions are possible to a certain location. The *switching sets* or *transition sets*[2] $T_{l,k} \subset G_l$ are given by

$$T_{l,k} = J_l^{-1}(S_k) = J_l^{-1}(\{k\} \times X_k).$$

Figure 3.9 shows a simple setup to exemplify the different sets of the GDHSA environment.

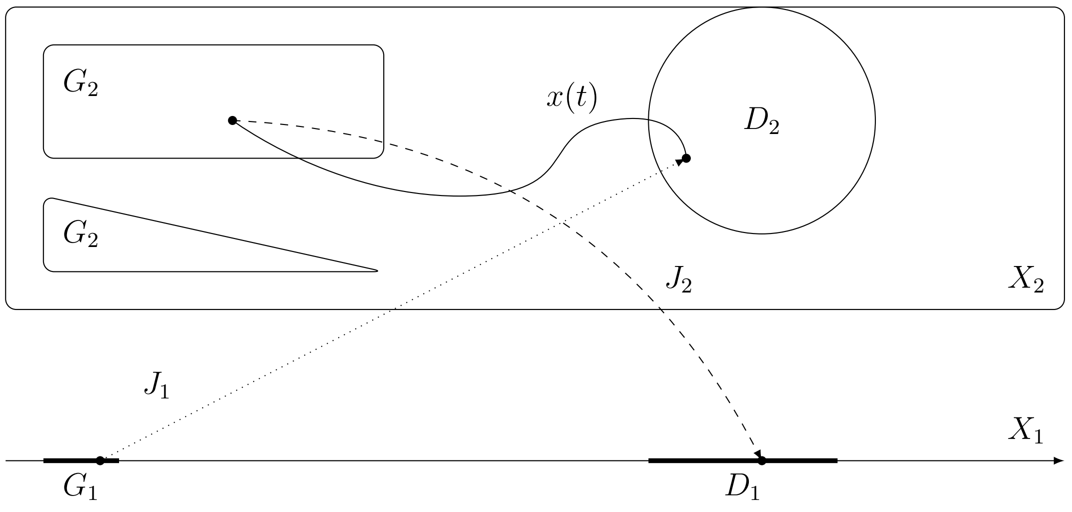


Figure 3.9: An Illustration of the Behaviour of a Generalised Dynamic Hybrid System Automaton for the Setup given by $L = \{l_1, l_2\}$, $\dim X_1 = 1$, $\dim X_2 = 2$ and some exemplary Configuration. For simple Notation the Location l_i is illustrated in the State Space, etc. simply by i.

Remark 3.16. According to Definition 2.6 the differential algebraic equation in each location l is given by a function $F_l\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^m$, which relates the vectors $\boldsymbol{x}\colon \mathbb{R} \to \mathbb{R}^n$ and $\boldsymbol{w}\colon \mathbb{R} \to \mathbb{R}^q$ via $F_l(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{w}) = 0$.
The generalisation $\boldsymbol{X} = \{X_l\}_{l\in L}$ and $\boldsymbol{W} = \{W_l\}_{l\in L}$ make it necessary to adapt this description of the dynamics in a location, because each F_l corresponds to a certain X_l and W_l. This results in the DAE $F_l(\boldsymbol{x}_l, \dot{\boldsymbol{x}}_l, \boldsymbol{w}_l) = 0$. For better readability of this thesis, the notation $F_l \in F_L$ will be kept, even though the notation in case of GDHSA implies a function $F_l\colon \mathbb{R}^{n_l} \times \mathbb{R}^{n_l} \times \mathbb{R}^{q_l} \to \mathbb{R}^{m_l}$, where $m_l \in \mathbb{N}$ and

$$n_l = \dim X_l, \qquad q_l = \dim W_l$$

is fixed for each $l \in L$. This means the domain of definition is related to the current dimension of the state space and the communication space, but still denoted by $F_l \in F_L$ to keep the intuitive understanding of the mathematical framework.

[1] π_1 denotes here the (algebraic) projection on the first component of $G(A) \cap S_l$, which is a set in S, thus the component in one of the X_l.

[2] In general manifolds are considered.

To complete the section of generalisation, the last step of including discrete state variables and controlled transitions are presented. Following [6], [5], as well as [19] and [32] the inclusion will be done by refinement of the generalised dynamic hybrid system automaton definition.

To cover the control issues in hybrid systems, the continues dynamics has to be reformulated in an input–output formalism. Therefore, it will be proceeded to split the vector of external variables as it is specified in Definition 2.7. To apply te DAE model either equation (2.11), for an implicit formalism, or equation (2.10), for an explicit formalism is used.

Definition 3.17 (Controlled Generalised Dynamic Hybrid System Automaton, CGDHSA)**.** Let L, $\boldsymbol{X}$, $\boldsymbol{G}$ and S be given as in Definition 3.15. Furthermore consider:

(a) The continuous dynamics in a state $l \in L$ is either given implicit by $F_l \in F_L$ or explicit by $(f_l, g_l) \in \mathcal{F}_L$, with the *continuous input* $\boldsymbol{u}_l \in U_l$, where U_l are *external continuous control sets* U_l collected in $\boldsymbol{U} = \{U_l\}_{l \in L}$.

(b) $\boldsymbol{V} = \{V_l\}_{l \in L}$ is the collection of *external discrete control sets* V_l, with the *discrete input* $v \in V_l$.

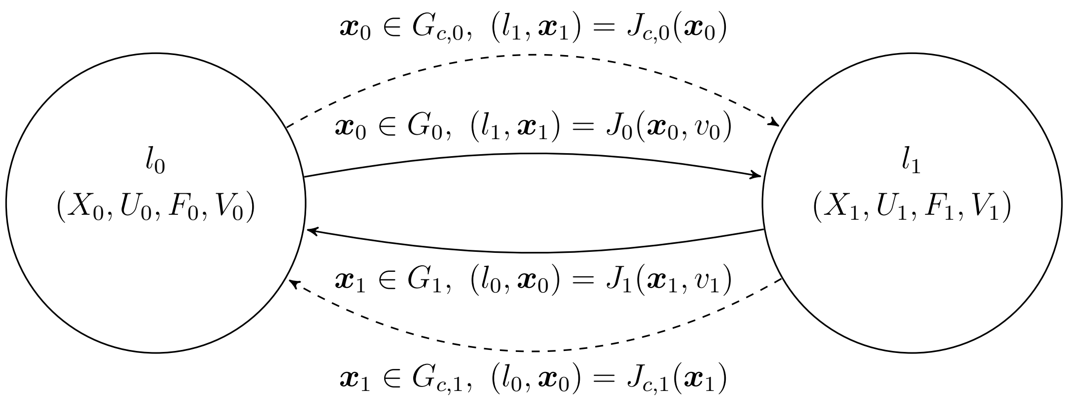
(c) $\boldsymbol{J} = \{J_l\}_{l \in L}$ is the collection of $J_l \colon G_l \times V_l \to S$ *autonomous jump transition maps*, controlled by the V_l.

(d) $\boldsymbol{G}_c = \{G_{c,l}\}_{l \in L}$ is a collection of *controlled jump sets* $C_l \subset X_l$.

(e) $\boldsymbol{J}_c = \{J_{c,l}\}_{l \in L}$, where $J_{c,l} \colon G_{c,l} \to \mathcal{P}(S)$, is the collection of *controlled jump destination maps.*

The nonuple[3] $(L, \boldsymbol{X}, \boldsymbol{U}, F_L, \boldsymbol{G}, \boldsymbol{J}, \boldsymbol{V}, \boldsymbol{G}_c, \boldsymbol{J}_c)$, respectively $(L, \boldsymbol{X}, \boldsymbol{U}, \mathcal{F}_L, \boldsymbol{G}, \boldsymbol{J}, \boldsymbol{V}, \boldsymbol{G}_c, \boldsymbol{J}_c)$ is called a *controlled generalised dynamic hybrid system automaton* $\mathcal{H}_c$.

For CGDHSA there is as well an automaton representation available, transitions are divided in autonomous transitions which has to be taken and controlled transitions which can may be taken according to the external control command. An illustration of a simple situation of two locations are illustrated in Figure 3.10.



$\boldsymbol{x}_0 \in G_{c,0},\ (l_1, \boldsymbol{x}_1) = J_{c,0}(\boldsymbol{x}_0)$
$\boldsymbol{x}_0 \in G_0,\ (l_1, \boldsymbol{x}_1) = J_0(\boldsymbol{x}_0, v_0)$
l_0
(X_0, U_0, F_0, V_0)
l_1
(X_1, U_1, F_1, V_1)
$\boldsymbol{x}_1 \in G_1,\ (l_0, \boldsymbol{x}_0) = J_1(\boldsymbol{x}_1, v_1)$
$\boldsymbol{x}_1 \in G_{c,1},\ (l_0, \boldsymbol{x}_0) = J_{c,1}(\boldsymbol{x}_1)$

Figure 3.10: Automaton Representation of a Controlled Generalised Dynamic Hybrid System for two Locations. For a simple Notation the Location l_i is illustrated by an Index $i = 0, 1$.

[3]The formal name of a 9–tuple.

The controlled environment in GDHSA enables to implement transitions which, depending on external controls, may be taken or not. This means the continuous trajectory can pass such a guard set without executing the jump. To demonstrate a certain scenario Figure 3.11 shows a simple setup to exemplify the dynamics of the GDHSA environment.

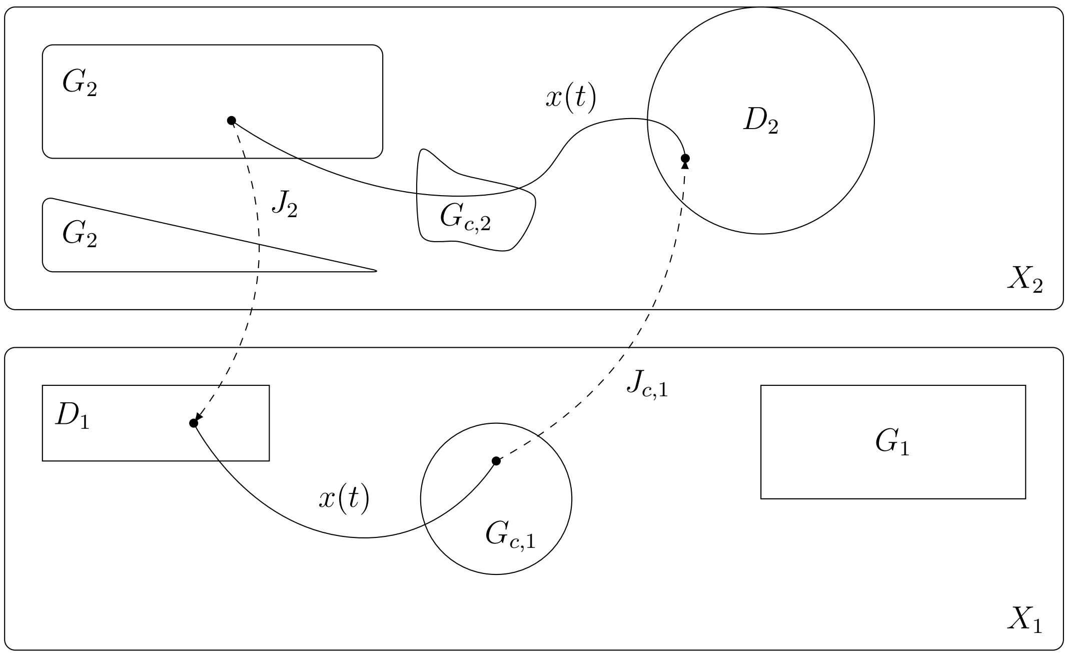


Figure 3.11: An Illustration of a Controlled Generalised Dynamic Hybrid System for the Setup given by $L = \{l_1, l_2\}$, $\dim X_1 = \dim X_2 = 2$ and some exemplary Configuration. For simpler Notation, the Location l_i is illustrated in the State Space, etc. simply by placing an Index i.

Equipped with this mathematical environment, the discussion of the architecture of an hybrid system per se, can be concluded. In the end of section 3.3 the system architecture of a controlled hybrid system is illustrated in Figure 3.8. For a definition of the whole system concept, a fitting mathematical environment is missing. Bringing this two aspects together allows the

Definition 3.18 (Controlled Dynamic Hybrid System, CDHS)**.** A system architecture as illustrated in Figure 3.8 provided with the mathematical environment of a CGDHSA in Definition 3.17 is called a *controlled dynamic hybrid system.*

Chapter 4

State Event Handling in the Numerical Environment

This chapter is dedicated to the handling of state events in a simulation environment. The simulation environment, in a first consideration, will contain the system description for a certain dynamics and the numerical tasks and problems which has to be resolved during the simulation run. Figure 4.1 illustrates this simplified environment structure.

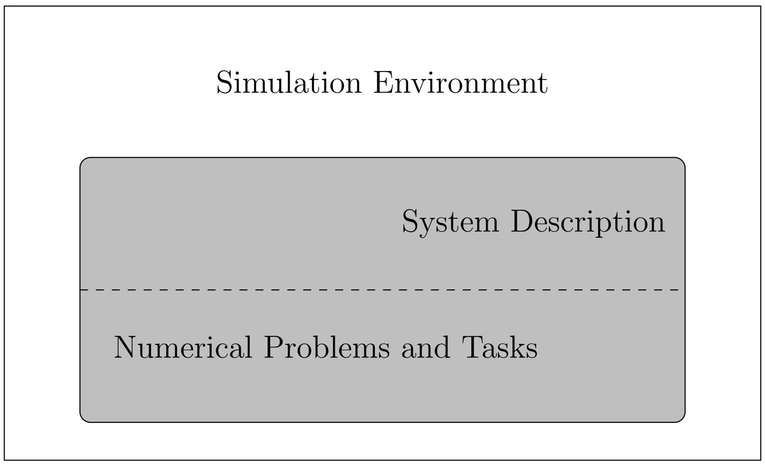


Figure 4.1: Simplified Structure of a Simulation Environment for Dynamical Systems.

The chapter follows the chronology in this summary. First, a classical simulation run of a DAE model will be considered. This results in the simulation loop, which illustrates the procedure of this kind of dynamical systems. Afterwards, this structure will be adapted to state event models and the corresponding simulation run. This section will cover as well a deep analysis of state event models and the handling in simulation environments. In this setup, will be also covered the numerical aspect of handling in the simulation environment. State event finding algorithms will be addressed, where traditional algorithms will be referred and one application of a certain method will be introduced. The final section of this chapter will cover the possibility of simplifications of the numerical framework for certain model structures.

4.1 Simulation Environment Structure

4.1.1 Classical Dynamical Models

This section will discuss the simulation structure for a DAE model. The model is assumed to be given through equation (2.10), where an explicit ODE and an algebraic equation defines the model. The procedural steps for the simulation run concerns the numerical computations for the model equations and the numerical integration. The simulation loop consists of three circles which are responsible for different computations regarding the model equations or the numerical solving algorithms. These are the followings:

(a) Model containing algebraic loops: Interaction between the dynamic and the algebraic equation.

(b) Numerical ODE solver for fixed step size and order simulation:

   (a) Model containing algebraic loops

   (b) Numerical ODE solver

   (c) Iteration Algorithm

(c) Error controlled simulation

   (a) Numerical ODE solver, including model containing algebraic loops

   (b) step size and order control

The mentioned iteration algorithm is used for the model containing algebraic loops and also in the numerical ODE solver algorithm. Mostly it is a form of the Newton iteration, but in general it can be an arbitrary numerical iteration. For a better presentation the three circles in the simulation loop are illustrated in Figure 4.2.

The figure illustrates tasks which are required for the simulation run, following the chapter about ”Differential Algebraic Equation Solvers” in [10]. The different circles separate different thematic regions which are interacting at certain points. Following the structure, the first task is inside the circle of model containing algebraic loops. There the issue of algebraic loops will be considered and especially the connection of the differential and algebraic model description in the state equation is discussed. Basically, the dynamics is given via the ODE part of the model description, the algebraic equation is responsible for the connection of the algebraic variables and the state variables. If the state vector is known and respectively the parameter and input vector, the algebraic vector is computed by the algebraic equation. The next circle brings the ODE solver in considerations, which is responsible for the time progress of the state vector. To get back to the model description in equation (2.10), the numerical considerations can be analysed.

For the following considerations for the numerical simulation, a decomposition $\mathcal{Z}$ is assumed for the fixed simulation time interval $[t_0, T]$. That means $\mathcal{Z} = \{t_0, t_1, \dots, t_k, t_{k+1}, \dots, T\}$ is the discrete set of moments for which the simulation algorithms derives values of the considered variables. The assumed continuous values are in the simulation environment only available at the time instances of the so-called grid $\mathcal{Z}$. The state vector $\boldsymbol{x}(t_k)$ is denoted as $\boldsymbol{x}_k$, similar

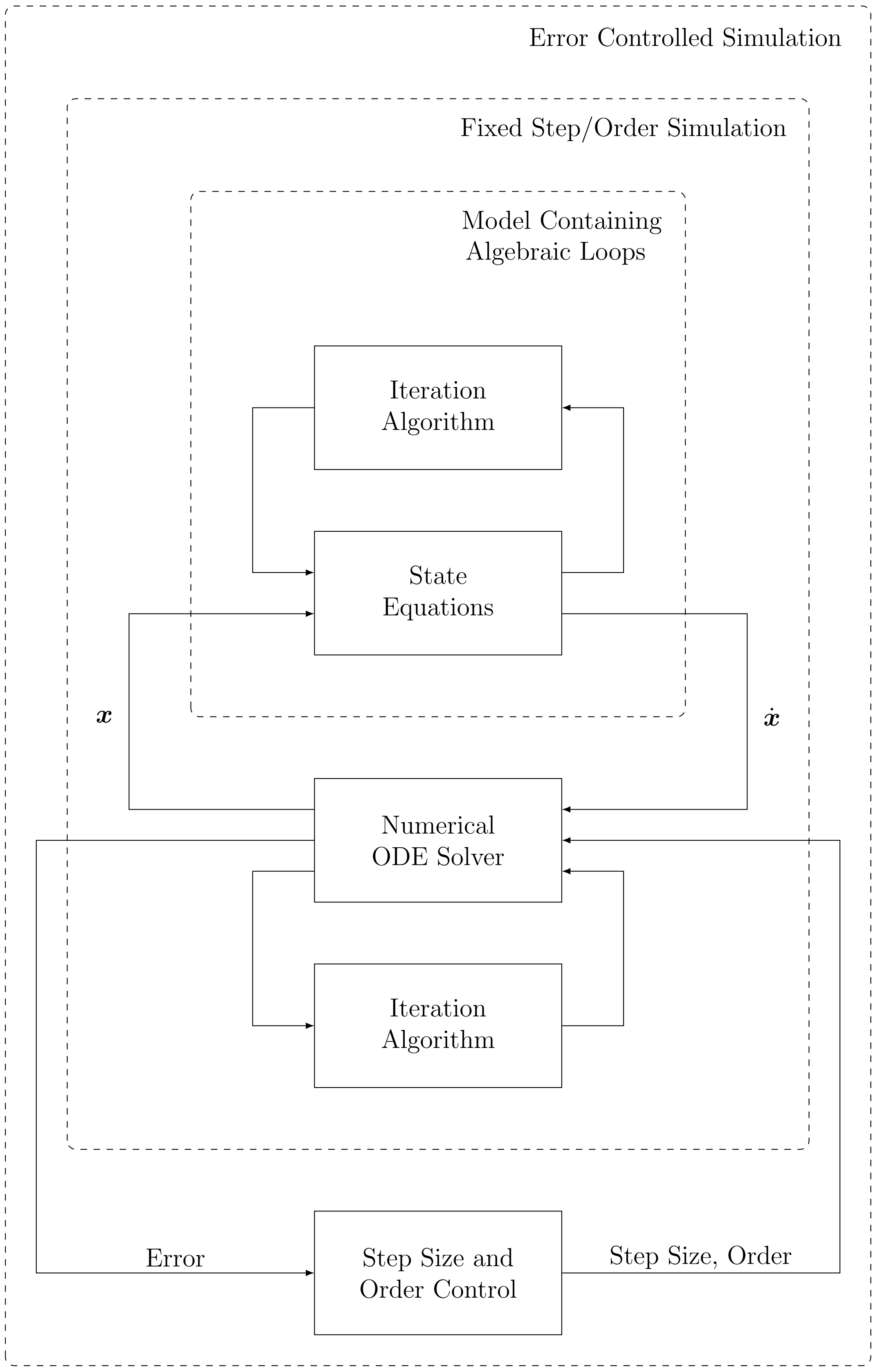


Figure 4.2: Illustration of the Simulation Loop of a Dynamical System containing the three Circles.

for all other involved vectors. The algebraic equation from equation (2.10) is written in the notation

$$g(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}, t_k) = \boldsymbol{0},$$

where $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ is one certain value of the given input vector. The parameter vector $\boldsymbol{p}$ is assumed to be constant for all times. For given $\boldsymbol{x}_k$ the current value of the algebraic variables can be calculated.

Remark 4.1. This calculation of the algebraic variables may concern a numerical iteration, which is covered by the inner circle in the simulation environment and is called the dissolution of algebraic loops.

The calculation of the algebraic variables requires that the state vector is known. This issue is covered by the ODE solver, which calculates, in each time step, the current value of the numerical approximation $\boldsymbol{x}_k$. Assuming the explicit Euler method for this numerical computation the state equation results in

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h_k} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}, t_k),$$

where $h_k = t_{k+1} - t_k$ denotes the step size. This approach for the explicit Euler method guides to the iteration

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}, t_k).$$

Starting from an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ the iteration together with the algebraic equation

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}, t_k) \tag{4.1}$$

$$g(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}, t_k) = \boldsymbol{0} \tag{4.2}$$

results in $\boldsymbol{x}_{k+1}$. The next step is the calculation of $\boldsymbol{z}_{k+1}$ from the algebraic equation and with it the calculation of $\boldsymbol{x}_{k+2}$, etc. The input vector is always known, therefor all $\boldsymbol{u}_k$ too. This procedure is done by the ODE solver, represented in the circle of fixed step/order simulation.

The outer circle in the simulation loop takes care on numerical quality criteria. The error (estimation) of the ODE solver controls the step size and order control of the simulation loop. In this thesis this issue will be not covered, being that the focus lies on the model description and characterisation of state events.

4.1.2 State Event Models

For state event models, the simulation procedure has to be adopted, as published in [34]. The formulation of a hybrid system has to be given in the understanding of a simulation environment point of view. In a hybrid automaton, the location change is associated with a transition to another location. In terms of the simulation environment, not the environment jumps in another location, the corresponding dynamic of the environment will be updated. This update has to be done if a certain indicating device is requiring this. This device, in the given framework, will be the event function as introduced in Definition 2.22. The computation of one more function in the environment is required and the procedure of a simulation run has to be stopped if an event occurs.

Subsection 2.2.3, State Event Procedure, introduce the procedural basics of state event handling and the particular steps has to be covered by state event handling algorithm. In the following paragraphs, the model description of a dynamic hybrid system will be incorporated in the simulation structure and state event handling will be discussed on basis of this structure in detail in the following section.

The starting point is mainly the same environment as introduced in subsection 4.1.1. The starting model description is in a first consideration as well given by a equation (2.10). To focus on the procedural aspect of the simulation environment, this model description is called $\mathcal{M}$. The major additional tasks which have to be fulfilled in the simulation environment with state event model are:

(a) State Event Detection: The detection of a state event results in stopping the simulation run, in particular the ODE solver.

(b) State Event Localisation: After detection of the state event the accurate event time t^* has to be localised.

(c) State Event Action: At the localised event time the model description $\mathcal{M}$ has to be changed to a new model description $\mathcal{M}_{\text{new}}$.

The here listed additional tasks and the numerical tasks are combined illustrated in Figure 4.3.

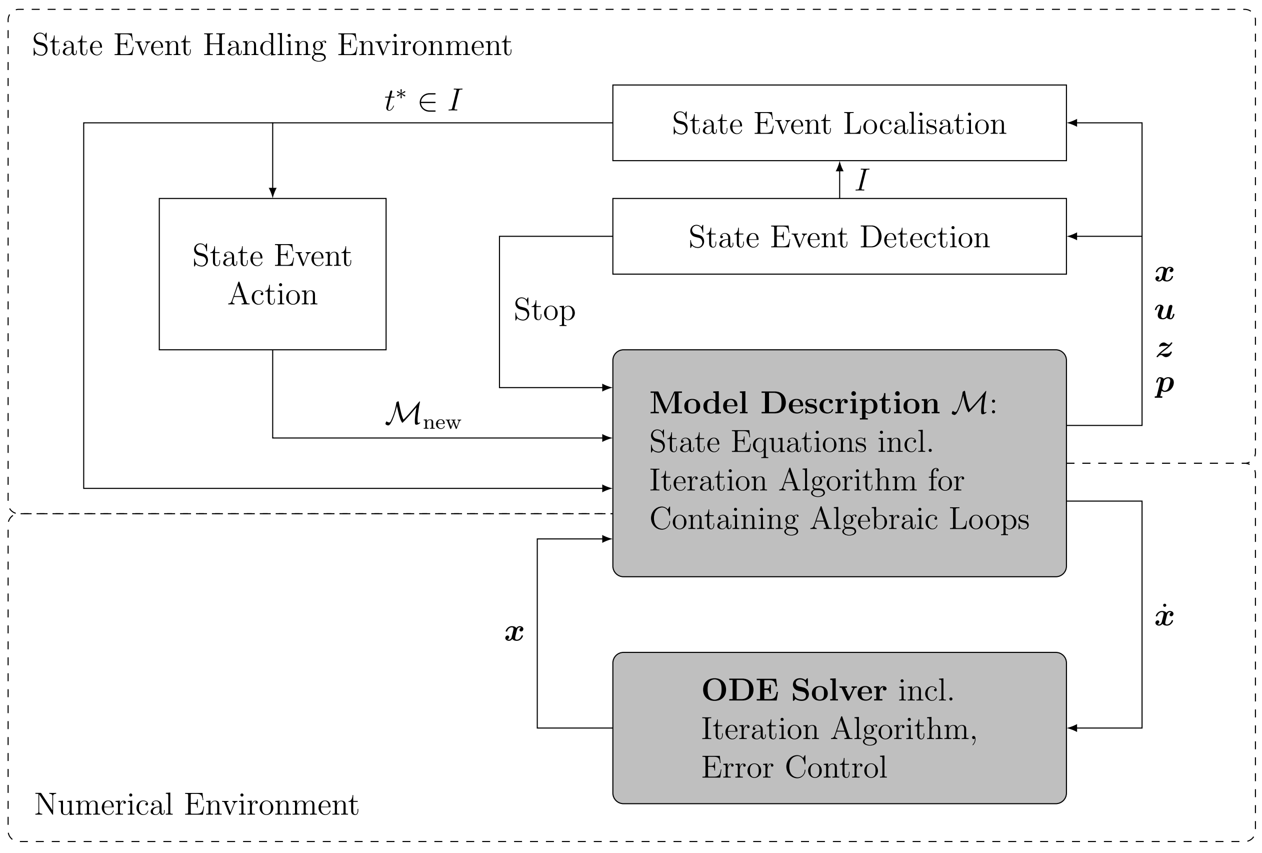

State Event Handling Environment
$t^* \in I$
State Event Localisation
I
State Event Detection
State Event Action
Stop
$\boldsymbol{x}$
$\boldsymbol{u}$
$\boldsymbol{z}$
$\boldsymbol{p}$
$\mathcal{M}_{\text{new}}$
Model Description $\mathcal{M}$: State Equations incl. Iteration Algorithm for Containing Algebraic Loops
$\boldsymbol{x}$
$\dot{\boldsymbol{x}}$
ODE Solver incl. Iteration Algorithm, Error Control
Numerical Environment

Figure 4.3: Illustration of the Simulation Loop Tasks for a State Event Model, consisting of the State Event Environment and the Numerical Environment.

4.2 State Event Handling Procedure

Based on Figure 4.3, the different tasks regarding the state event will be discussed and for this reason the mathematical framework defined in subsection 2.2.2, State Event Modelling, will be used and partially extended.

4.2.1 State Event Detection

In Figure 2.8, a simulation run with state event detection was in principle introduced. Having now the numerical aspects of a simulation run available, the state event detection can be discussed more in detail. Considering a time grid $\mathcal{Z} = \{t_0, t_1, \dots, t_k, t_{k+1}, \dots, T\}$ of the simulation interval $[t_0, T]$ the detection procedure can be applied as introduced in a survey in corresponding to Figure 2.9. In general the event function e defines the event time by the equation

$$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \boldsymbol{0}.$$

For simplification in this section it is assumed a scalar event function, this means the state event is defined by only one equation. Higher dimensional considerations would apply to the following discussions on each component of the vector valued event function. Because the numerical values $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, $\boldsymbol{z}_k$ and $\boldsymbol{p}$ are available, the chance to compute the exact event time is very low. According to this problem, one possibility would be to decrease the step size to reach, with a certain tolerance, the event time. But this would result in a higher computation effort and so in a longer duration of computation. The middle course is to detect the event and after the detection to localise the event time with a better tolerance.

An event occurs, if the sign of the event function at a certain time instance is different from the sign of the event function one time step afterwards. This corresponds to the condition

$$\operatorname{sign} e(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}) \neq \operatorname{sign} e(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{p}). \tag{4.3}$$

The result of an event detection is the interval $[t_k, t_{k+1}]$ in which the zero of the event function is contained. Figure 4.4 illustrates the setup for the scalar case.

4.2.2 State Event Location

Based on the event detection the event time, t^* has to be computed by an additionally numerical iteration. The state event location provides the interval, in which the zero of the event function is located, as illustrated in Figure 4.3 and is denoted as I and in the considerations of the subsection before, it is referred as $I = [t_k, t_{k+1}]$.

To handle the notation regarding the issues of the event function, the values of the event function e at a certain moment t_k are denoted as e_k for $k \in \{1, 2, \dots\}$. Here is referred Remark 2.27 which addresses exactly this issue.
An explicit dependency of the event function of time is only in the case of a possible time event. In case of a state event, the event function is depending on the state vector, on the input- and algebraic vector as well as on the parameter vector. The parameter vector is assumed to be

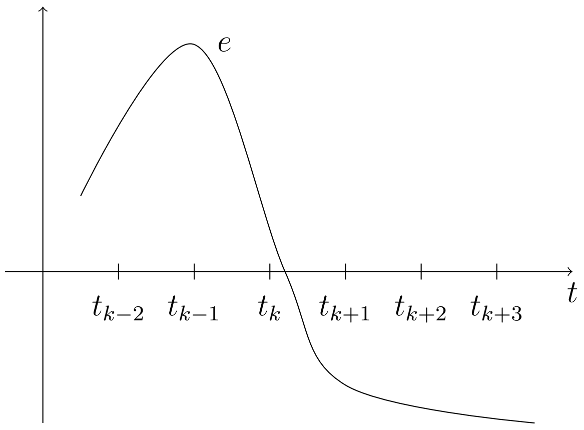


Figure 4.4: An exemplary Event Detection by detect the Zero of an Event Function e over the time Grid $\mathcal{Z} = \{\ldots, t_{k-2}, t_{k-1}, t_k, t_{k+1}, t_{k+2}, \ldots\}$.

constant and the input vector is as an external input known. The state and algebraic vector are defined via the differential algebraic equation and interfere with each other in the numerical computation as addressed in equation 4.1.

The facts discussed above leads to the basic ideas of *event location algorithm.* In principle there are two possible algorithms to locate the event time:

(a) Interpolative Algorithm

(b) Iterative Algorithm

Interpolative Algorithm. An interpolative algorithm is working with the current available values of the event function e_k, respectively with the corresponding values of the state vector $\boldsymbol{x}_k$ and the algebraic vector $\boldsymbol{z}_k$. Based on this values, an interpolation function, mostly polynomials, can be constructed.

Definition 4.2. An *interpolative event function* $\hat{e}$ for state event location is defined as

$$\hat{e}(t) = \texttt{Interpol}(\boldsymbol{e}, \mathcal{Z}),$$

where $\boldsymbol{e}$ is the vector of the numerical values e_k and $\mathcal{Z}$ the basis time grid. `Interpol` is a certain used interpolative algorithm.

The simplest approach is the linear interpolation as $\hat{e}$. In this case the interpolative algorithm needs only e_k and e_{k+1} on $[t_k, t_{k+1}]$. For more sophisticated approaches, more numerical values of the event function are needed. This is respected in the definition of the interpolative event function by using the expression of an algorithm, where the list of parameters can be dynamic.

The event time t^* can be determined by finding the zeros of the interpolative event function,

$$\hat{e}(t) = 0. \tag{4.4}$$

As a second possibility, there are iterative algorithms which offer the possibility to iterate the event time. This approach requires iterative computations to locate the event time. For notation the algorithm `Iteration` will be formalised. The parameter list can not be given in general, because different iterations are working in a different way. The following paragraph will give an introduction of the classical numerical approaches, discussed in detail in [8] and [7].

Classical Iterative Algorithm. In order to enter in the field of iterative state event location algorithms, first classical methods will be introduced and discussed. Basically, an iterative method needs a certain tolerance, where the tolerance terminate. In the setup of locating the event time, the starting is from the time interval $[t_k, t_{k+1}]$ which is provided from the event detection. This initial interval will be refined up to a certain exactness, given by the tolerance of the iterative algorithm.

The first discussed algorithm is the *bisection method.* For the description of the algorithm, the starting point is the interval $[t_k, t_{k+1}]$ and the fact $\operatorname{sign} e_k \neq \operatorname{sign} e_{k+1}$. Without loss of generality assume $e_k < 0$ and $e_{k+1} > 0$. Set $a_1 := t_k$ and $b_1 := t_{k+1}$ and denote $\tilde{e}(\cdot)$ be the computation of the numerical value of the event function at a certain moment in the interval $[a_1, b_1]$. Following the algorithmic steps:

- Calculate $e_1 := \tilde{e}(\frac{a_1+b_1}{2})$.
  (a) If $e_1 > 0$: $a_2 := a_1$ and $b_2 = \frac{a_1+b_1}{2}$.
  (b) If $e_1 < 0$: $a_2 := \frac{a_0+b_0}{2}$ and $b_2 = b_1$.
  (c) If $e_1 = 0$: STOP, zero found.
- Calculate $e_2 := \tilde{e}(\frac{a_1+b_1}{2})$.
  (a) If $e_2 > 0$: $a_3 := a_2$ and $b_3 = \frac{a_2+b_2}{2}$.
  (b) If $e_2 < 0$: $a_3 := \frac{a_2+b_2}{2}$ and $b_3 = b_2$.
  (c) If $e_2 = 0$: STOP, zero found.
- And so forth.

From one iteration step to the next iteration step, the range of interval is halved. The iteration will terminate, if for a given tolerance `TOL` and a certain $j \in \mathbb{N}$, the condition

$$|a_j - b_j| < \texttt{TOL} \tag{4.5}$$

is fulfilled. Of course for each iteration step, the numerical environment has to provide the corresponding solutions for the state vector and the algebraic vector, which results in higher numerical costs. If the condition in equation (4.5) is fulfilled, the event time is located by

$$t^* = a_j.$$

Further on, the method is introduced which uses the zeros of the secant of the points (t_k, e_k) and t_{k+1}, e_{k+1} and start with this an iterative procedure. For a precise understanding, in general, this method would be called the *regula falsi method.* This method uses the secant on an interval, where the function values have different signs, whereas the secant method this restriction not require. The secant method uses two points in the neighbourhood of a function zero, where the function values can be probably with the same sign. This can be the starting point and as well, in a certain iteration step this situation can occur. Due to the foregone event detection procedure, which provides the interval $[t_k, t_{k+1}]$, where the event function has different signs, the methods are in the initial step equal. In the following algorithm description it will be required the condition of different signs and this means the regula falsi method is used in this event location iteration. For the formulation of the algorithm assume without loss of generality $e_k < 0$ and $e_{k+1} > 0$ and set $a_1 := t_k$ and $b_1 := t_{k+1}$. The next considered location is calculated as the zero of the corresponding secant

$$s(t) = \frac{e_{k+1} - e_k}{b_1 - a_1}\,(t - a_1) + e_k.$$

The zero of s results in

$$\hat{t} = a_1 - \frac{b_1 - a_1}{e_{k+1} - e_k}\,e_k. \tag{4.6}$$

Using this time value $\hat{t}$ to compare the sign of the event function $e(\hat{t})$ the interval can be shortened by replacing the interval border by this time value. Switching to this this new interval from the beginning, the iteration algorithm for the *secant method* can be given as follows.

Note 4.3. Again, the notation $e(\cdot)$ indicates the numerical approximation of the event function e at the mentioned time instance.

The j–th step of the iteration is calculated via

$$c_{j+1} = a_j - \frac{b_j - a_j}{e(b_j) - e(a_j)}\,e(a_j). \tag{4.7}$$

The value c_{j+1} is used according to the following regulations:

- If $e(c_{j+1}) = 0$: STOP, zero found.
- If $\operatorname{sign} e(a_j) = \operatorname{sign} e(c_{j+1})$: $a_{j+1} = c_{j+1}$ and $b_{j+1} = b_j$.
- If $\operatorname{sign} e(b_j) = \operatorname{sign} e(c_{j+1})$: $a_{j+1} = a_j$ and $b_{j+1} = c_{j+1}$.

The condition of termination is the same condition is applied as for the bisection method, given via equation (4.5).

Finally the most prominent method, the *Newton iteration* is introduced to locate the event time. The Newton method uses the zeros of the tangent of the point (t_k, e_k) and starts with it an iterative procedure. Of course the tangent requires the knowledge of the derivative, which has to be computed numerically. When this is provided, the tangent at the moment t_k is given by

$$T(t) = \dot{e}(t_k)(t - t_k) + e(t_k).$$

The zero of T results in

$$\hat{t} = t_k - \frac{e(t_k)}{\dot{e}(t_k)}. \tag{4.8}$$

Starting from this point using $t_k = a_1$, the iteration algorithm is given by

$$a_{j+1} = a_j - \frac{e(a_j)}{\dot{e}(a_j)}. \tag{4.9}$$

Remark 4.4. The computation of the numerical derivative requires more computational power. The simplification of this method is, to use the secant instead of the tangent, which results in the secant method. Due to the fact, that the Newton method does not require a certain condition regarding the sign, as well the secant method does not require this. This is the difference to the regula falsi.

The Newton iteration is the standard method used in system simulation. Either the general one or an adapted version is applied for the event time location. In section 4.3.1 this aspect will be picked up again and some structural simplification potentials will be presented.

Figure 4.5 illustrates the three presented methods for the initial step, to give a quantitative impression of the construction.

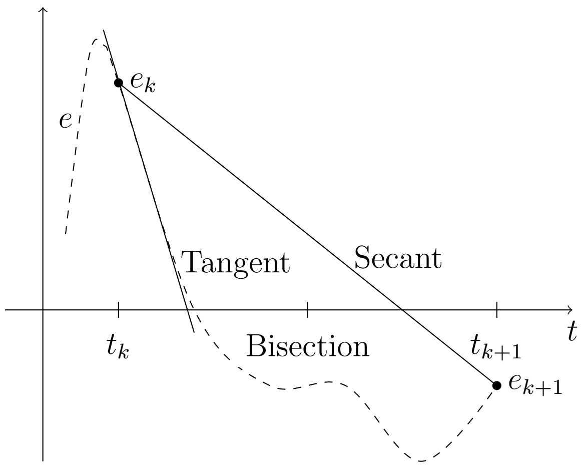


Figure 4.5: Comparison of different classical Methods for the Location of the Zeros of an Event Function e in the Interval $[t_k, t_{k+1}]$.

This paragraph summarises the classical methods for state event localisation. An extraordinary method will be introduced in the next paragraph.

Henon Method. An alternative method of numerical iteration for state event finding is presented in [8]. This method has the origin in astrophysics, several decades before state event finding was an issue, for example [24]. To introduce the method of Henon a dynamical system of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix} = f(\boldsymbol{x}),$$

as introduced in [7] and [12] is used. Consider the first equation of the system

$$\frac{\mathrm{d}}{\mathrm{d}t}x_1 = f_1(\boldsymbol{x})$$

and change the independent variable to

$$\frac{\mathrm{d}t}{\mathrm{d}x_1} = \frac{1}{f_1(\boldsymbol{x})}.$$

Concerning the chain rule applied on the second component results in

$$\frac{\mathrm{d}}{\mathrm{d}t}x_2 = \frac{\mathrm{d}x_2}{\mathrm{d}t}\frac{\mathrm{d}x_1}{\mathrm{d}x_1} = \frac{\mathrm{d}x_2}{\mathrm{d}x_1}\frac{\mathrm{d}x_1}{\mathrm{d}t} \qquad \text{and leads to} \qquad \frac{\mathrm{d}x_2}{\mathrm{d}x_1} = f_2(\boldsymbol{x})\frac{1}{f_1(\boldsymbol{x})}.$$

Following this strategy the resulting system is given by

$$\frac{\mathrm{d}}{\mathrm{d}x_1}\begin{pmatrix} t \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{1}{f_1(\boldsymbol{x})} \\ \frac{f_2(\boldsymbol{x})}{f_1(\boldsymbol{x})} \\ \vdots \\ \frac{f_n(\boldsymbol{x})}{f_1(\boldsymbol{x})} \end{pmatrix}. \tag{4.10}$$

The change of the independent variable principle can be used for localisation of the state event. This is done via the usage of the Henon Method in relation to poincare maps. Poincare maps are is called the intersection of an orbit of a dynamical system and a lower-dimensional subspace. As presented in [8], the subspace is represented by a function $H\colon \mathbb{R}^n \to \mathbb{R}$ via

$$H(\boldsymbol{x}) = H(x_1, x_2, \ldots, x_n) = 0. \tag{4.11}$$

This function H represents a certain structure of an event function e, which depends only on the state vector.

Consider the following simplified subspace of type

$$H(x_1, x_2, \ldots, x_n) = x_k - a = 0.$$

For $k = n$ the this equation, which is a special case of a event function, is related to the solution of the differential equation system (4.10), where the variables $x_1, x_2, \ldots, x_{n-1}, t$ are depending on the independent variable x_n. The system (4.10) is in general much more complex and the division through $f_n(\boldsymbol{x})$ might cause problems, but it allows numerical computation of the intersection of the dynamical system with the subspace defined by (4.11). Therefore, following numerical strategy is applied :

(a) Numerical solution of the original dynamical system, until t_k, the moment just before the intersection with $H(\boldsymbol{x}) = x_n = a$ and stopping the simulation at $x_n(t_k) \neq 0$.

(b) Numerical solution of the modified system of equation (4.10) by the same ODE solver starting from $x_n(t_k)$ with the step size $\pm x_n(t_k)$, meeting the intersection almost exactly.

The subspace $H(\boldsymbol{x}) = 0$ can be interpreted now as set of zeros of the event function e, so that the above algorithm can be used as alternative method for state event location without any interpolation and iteration.

For more complex subspaces $H(\boldsymbol{x}) = 0$, than specified before, the method can be generalised in the following sense. The subspace will be formulated via a differential equation using the function

$$x_{n+1}(t) = H(x_1, x_2, \ldots, x_n)$$

and compute the derivative according to the chain rule to

$$\frac{\mathrm{d}}{\mathrm{d}t}x_{n+1}(t) = \sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}x_i}H(x_1, x_2, \ldots, x_n)\frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}x_i}H(x_1, x_2, \ldots, x_n) f_i(x_1, x_2, \ldots, x_n).$$

The Henon method can be used for event time location. After the event detection, the interval $[t_k, t_{k+1}]$ is provided. The event time $t^* \in [t_k, t_{k+1}]$ is determined by numerical solving of the system

$$\frac{\mathrm{d}}{\mathrm{d}x_{n+1}} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ t \end{pmatrix} = \frac{1}{f_{n+1}(x_1, x_2, \dots, x_n)} \begin{pmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \\ 1 \end{pmatrix}, \tag{4.12}$$

while f_{n+1} is given by

$$f_{n+1}(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}x_i} H(x_1, x_2, \dots, x_n) f_i(x_1, x_2, \dots, x_n).$$

It is necessary that $f_{n+1}(x_1, x_2, \dots, x_n) \neq 0$ on $(t_k - \epsilon, t^* + \epsilon)$ and the ODE solver is operating on the interval $[x_{n+1}(t_k), 0]$ with $x_{n+1}(t_k) = H(\boldsymbol{x}(t_k))$ and a step size of $\pm x_{n+1}(t_k)$. This results in the zero of the event function

$$t^* = t(x_{n+1}) = t(0),$$

where $t(\cdot)$ is the time function dependent on x_{n+1} after the change of depending variables.

The Henon method is not used as a stand alone, it is combined for the last iteration step after the event detection. An interesting link is given to the traditional state event location methods, via the equivalence

$$x = \dot{e}(t_k)(t - t_k) + e_k \qquad \Longleftrightarrow \qquad t = \frac{1}{\dot{e}(t_k)}(x - e_k) + t_k. \tag{4.13}$$

The Newton iteration in the time domain uses the same line as the Euler method for ODE solver in the state space, illustrated in the scalar case in Figure 4.6.

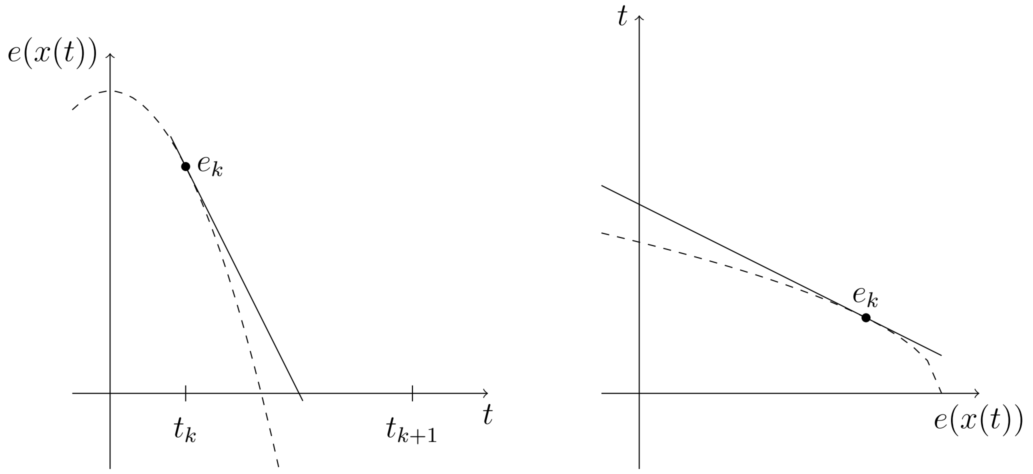


Figure 4.6: Equivalence of Newton Iteration in the Time Domain and Euler Method for ODE Solver in the State Space Domain.

Discontinuity Sticking. This effect is a numerical phenomena, which occurs in the process of state event location. The location needs the evaluation of the event function all over the interval $[t_k, t_{k+1}]$. After location of the event time, the discrete transition takes place and the state equations and algebraic equation is initialised for the next location activities. One problem which can occur in this setup is, as discussed in [41], that the new initialisation of the continuous dynamics influences as well the event function in such a way, that the event function is initialised with the correct values and it is obvious that the event time was wrongly located. The reason is an interpolative algorithm which delivers wrong values and so the found event time is wrong by itself. Figure 4.7 shows this effect and illustrates, that this effect occurs after the event was located.

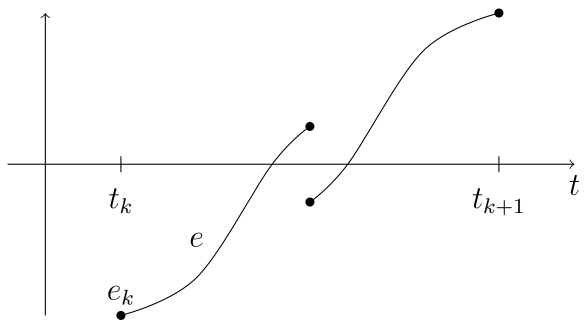


Figure 4.7: Illustration of the Discontinuity Sticking of an Event Function e over $[t_k, t_{k+1}]$.

4.2.3 State Event Action

After dealing with the numerical issues of state event detection and location, the last step in the state event handling procedure is the event action. The event action is illustrated in Figure 4.3 as providing the new simulation model $\mathcal{M}_{\text{new}}$ for the model description module of the simulation environment. This new model is for the environment, which is at this moment in the stopped simulation run, the update of the model description. Updates regarding several issues are possible, which are subdivided in the following listing:

- Parameter Change: This action means, that one parameter vector is either replaced by another one, or it may can change the size of the vector.
- Input Change: This case is equivalent to the parameter change, because the parameters can be interpreted as inputs and vice versa.
- State Vector Change: In this case the values of the state vector change.
- Derivative Vector Change: One or more of the components of the derivative functions change.
- Algebraic Equation Change: The equation which covers the algebraic part of the model description is changing.
- Model Change: This change is the most rigorous change in the model description. The state vector may change in dimension and description, initial conditions have to be provided for the change, etc.

In most of the cases, the state event action is related to some algorithmic descriptions of the changes in a particular case, but no mathematical framework is provided for this change. The classification is a first step to understand in which element the change is launched, the mathematical equipment is still missing.

An additional aspect, which is up to now not discussed, is a change or an action which is related to the output of the system. Here the variety last from a numerical issue to synchronise the time grid of the numerical computations to a certain moment where the output has to be provided, up to a change in the output equation, which concerns the model structure.

These listed and discussed issues of the event actions has to be embedded into a mathematical environment to describe the state event action in an abstract meaning. This allows a discussion about state event model on a general level without requiring a simulation model, where the change is given on a algorithmic level.

Chapter 5, Mathematical Characterisation of State Events, will introduce this mathematical framework. It is based on the previous chapters, which prepared the mathematical issues on a proper level. Moreover, the chapter will bring together the system architecture of a dynamical hybrid system with the simulation environment architecture and discuss potential influences.

4.3 Remarks regarding the Numerical Environment

4.3.1 Simplifications in the Numerical Environment

This section involves simplifications in the considerations of the numerical environment. The Aim is, to observe some special situations, where the given structure is providing benefits regarding the numerical computations, in order to reduce their number. The focus is oriented to the event function e. Even in the above introduced Henon method it was the structure of the event function, which offered a method to localise the state event with an efficient algorithm.

Proper event functions, for simplifications, are event functions which are depending only on state values, which are so-called autonomous event functions, issued in [34].

Definition 4.5. Consider an event function e, which is a case of the general definition given in Definition 2.22, structured as

$$e(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}) = \overline{e}(x_1, x_2, \ldots, x_n) - a = 0,$$

with a constant $a \in \mathbb{R}$, representing a certain threshold, offset, or similar. The function $\overline{e}\colon \mathbb{R}^n \to \mathbb{R}$ is called a *autonomous scalar event function.*

For this kind of event functions, it is possible to refer structure of the function, for which the numerical environment experiences simplifications. Assume a autonomous scalar event function in a pretty simple form

$$\overline{e}(x_1, x_2, \ldots, x_n) = x_k. \tag{4.14}$$

This represents the case, that the event is defined and that a certain state value is reaching a given bound. In cases of thresholds and offsets, this case is quite often to discover in engineering and information science. The numerical iteration of the event location, which of mostly a Newton iteration, can be simplified in this situation. The basic idea for the iteration method is given in equation (4.8) therefore, the derivative with respect to the time is necessary which in case of equation (4.14), is computed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{e}(x_1, x_2, \dots, x_n) = \dot{x}_k.$$

The benefit is, the value $\dot{x}_k$ is already available, because inside the iteration this value is needed for the computation of the value e_k, as illustrated in Figure 4.5 and Figure 4.3. This introducing idea can be generalised to *linear autonomous scalar event function*

$$\overline{e}(x_1, x_2, \dots, x_n) = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n, \tag{4.15}$$

where the derivative is given as

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{e}(x_1, x_2, \dots, x_n) = \alpha_1 \dot{x}_1 + \alpha_2 \dot{x}_2 + \dots + \alpha_n \dot{x}_n.$$

The computation requires only the evaluation of the linear combination of the available time derivatives. Due to excellent performance of numerical environments w.r.t. polynomials, it is worthy as well to analyse *polynomial autonomous scalar event function* defined as follows

$$\overline{e}(x_1, x_2, \dots, x_n) = \sum_{\alpha} a_\alpha \boldsymbol{x}^\alpha, \tag{4.16}$$

where the notation of multivariate polynomials is used. These are defined as linear combinations of monomials, this means

$$\boldsymbol{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n},$$

where $\alpha_i \in \mathrm{N}_0$. The computation of the values of the derivatives is simple, since the derivatives of the state values are known by the environment, which results in less computation cost.

Of course the analysis of this simplifications can be promoted further, but the simplifications in terms of computation costs will result in nothing more. The basic issue is the autonomous event function, which does not depend on algebraic variables or similar but only on the state variables itself.

4.3.2 Significance of Time Events

The thesis addressed mainly state events and neglected time events. Time events are introduced as a certain variant of the event function, where the time is not defined via a particular constellation of the state values, the time is direct fixed via the event function. For this situation the term time event is used. The question is; is it in the modelling concept of dynamic hybrid model useful to use this type of event and what can be a certain event action?

The main focus of time events is located in the numerical environment, not in the structural considerations of a model transition. A possible situation for time events can be, that the values of the output vector have to be known at a certain point in time. For this requirement a time event defines the event time, which is known in advance and is respected in the numerical environment. In particular the step size of the ODE solver has to be synchronised with the time event.

In this case the whole iteration of the event time is obsolete, because it is known in advance. The whole numerical environment, each component which uses a step size can synchronise to this given time and the simulation run does not have to be interrupted in order to find a certain event time.

To sum up, the time events are important for the numerical operations in a simulation run and belong to the numerical problems and tasks of Figure 4.1. This is part of the numerical environment and is not covered in the state event characterisation.

Chapter 5

Mathematical Characterisation of State Events

The structure of the thesis, up to now, followed the idea of constructing a mathematical characterisation for state events. Therefor, a mathematical environment was built and adopted up to the point, to be suitable for dynamic hybrid systems. In addition, as well the numerical framework was considered to integrate the numerical needs of a simulation environment for state event modelling in the whole description. Finally, this chapter will include the prepared ingredients to subsume a mathematical characterisation of state event modelling.

For this purpose, the structure of Figure 4.3 has to be adapted again. All whole tasks which are oriented to the numerical aspects, will be covered through a numerical framework and a second framework is covering the model description. Of course, the two parts are interacting and effecting each other. Figure 5.1 illustrates this structure.

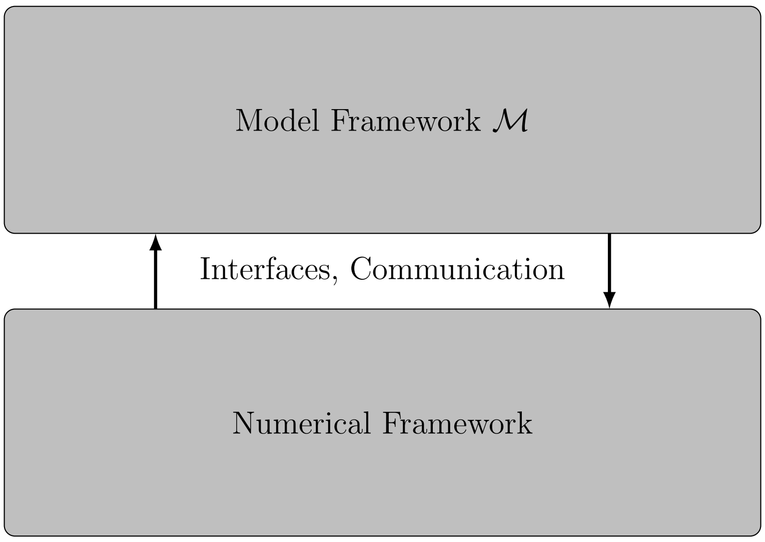


Figure 5.1: Illustration of the Framework used for State Event Characterisation.

The numerical framework in Figure 5.1 covers the numerical tasks from chapter 4, State Event Handling in the Numerical Environment. In the model framework, the mathematical description of a controlled input output hybrid system will be incorporated and completed by the output equation, which so far, is not present. This shows as well, that the model $\mathcal{M}$ contains much more than equations, as well the, so-called, discrete dynamics has to be covered there. The interfaces and communication ports indicate that the model description by itself is a mathematical device and the particular computations, of course, were handled in the numerical framework. For characterisation of state event models, the abstract mathematical description is indispensable.

As mentioned before in several situations, the characterisation of state events is done in an abstract mathematical model to make sure, this characterisation are not dependent on the particular implementation in a certain simulation environment. This would open the field to formulations which are more oriented to restrictions given through the simulation environment, than formulations which accommodate needs and duties from the system theory. This approach will worked out in the following sections. The different state event types will be addressed and formulated by using the mathematical environment of dynamic hybrid systems.

For the characterisation, the model framework has to be redefined for a proper formulation of the different types of state events. As a last step, the output equation has to be included in the model description.

Definition 5.1. Consider a state $l \in L$ of a hybrid system. Denote $(f_l, g_l, h_l) \in \mathcal{D}_l$ the *continuous dynamics* in a state $l \in L$.

Definition 5.2 (Controlled Generalised Input Output DHSA, CGIODHSA)**.** Consider the definition of a controlled generalised dynamic hybrid system automaton, given in Definition 3.17, with the following redefinition.
The continuous dynamics, in a state $l \in L$, is given by $(f_l, g_l) \in \mathcal{F}_L$, with the *continuous input* $\boldsymbol{u}_l \in U_l$, where U_l are *external continuous control sets* collected in $\boldsymbol{U} = \{U_l\}_{l \in L}$ and the *continuous outputs* $\boldsymbol{y}_l \in Y_l$, where Y_l is out of the collection of $\boldsymbol{Y} = \{Y_l\}_{l \in L}$, which are given by $\boldsymbol{y}_l = h_l(\boldsymbol{x}_l, \boldsymbol{u}_l, \boldsymbol{z}_l, \boldsymbol{p}_l, t)$. In each location, a parameter vector $\boldsymbol{p}_l \in \mathbb{R}^{r_l}$ is valid.

According to the fact, that the GDHSA represents a generalisation for a DHSA, also the DHSA has to be redefined.

Definition 5.3 (Input Output Dynamic Hybrid System Automaton, IODHSA)**.** Consider the definition of a dynamic hybrid system automaton, given in Definition 3.2, with the following redefinition.
The mapping Act: $L \to \mathcal{D}_L,\ l \mapsto (f_l, g_l, h_l)$ is a mapping that assigns to each location $l \in L$ a set of differential and algebraic equations as well as an output equation $(f_l, g_l, h_l) \in \mathcal{D}_L$, relating the state vector $\boldsymbol{x}$, the derivative of the state vector $\dot{\boldsymbol{x}}$, the algebraic vector $\boldsymbol{z}$ and the parameter vector $\boldsymbol{p}$ via

$$\begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0} \\ \boldsymbol{y} = h_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases} \tag{5.1}$$

The solutions of these (explicit) differential and algebraic equations are called the *activities* of the location l.

Definition 5.2 represents the most general definition of a DHSA, where it is possible to model all changes in the model structure. For some model actions, the DHSA is sufficient especially if the state space is not changing.

This definitions allows to understand the model framework $\mathcal{M}$ from Figure 5.1 either as an input output dynamic hybrid automaton or as a controlled generalised input output dynamic hybrid automaton. The state event modeling approach results in a corresponding change of the state in the automaton. The interconnection to the numerical environment hand over the numerical computations, iterations and as well stopping and restarting of the simulation run is covered in the numerical framework. Figure 5.2 shows a refined illustration, where the (generalised) dynamic hybrid system automaton is represented only by the continuous dynamics and labeled edges.

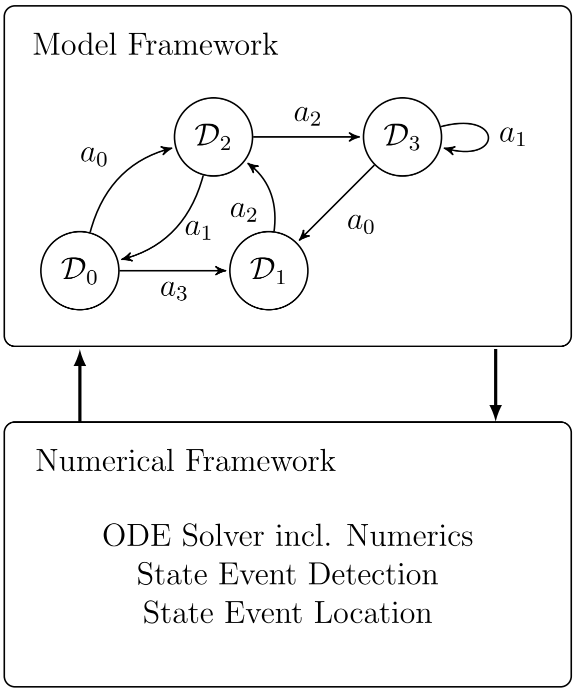


Figure 5.2: Illustration of the Setup for State Event Models in a Simulation Environment.

In the following sections the different types, as listed in subsection 4.2.3, of state events will be formulated in the given mathematical environment. Several examples will underline this mathematical formulation and illustrate with applications.

The whole model framework of a dynamic hybrid system is denoted by $\mathcal{M}$, as mentioned in the beginning of the chapter. Each of the locations of the hybrid automata is one sub–environment $\mathcal{M}_l$, which provides, beside the continuous dynamics, also the sets and mappings which describes the discrete dynamics. So $\mathcal{M}$ is the collection of all sub–environments $\mathcal{M}_l$, $l \in L$. Furthermore, in each location the dynamics $\mathcal{D}_l$ is given by the corresponding differential, algebraic and output functions. Due to the dependency of the functions by parameters and inputs, the formalism distinguishes the dynamics $\mathcal{D}_l$ and the corresponding equations $\mathcal{E}_l$, which involves as well parameters and inputs.

5.1 State Event Value Changes

State event actions can be characterised in two categories, value change and structural change. The following subsections discuss the value changes of several vectors and concludes the combination of these changes.

5.1.1 Change of Parameters and Inputs

This section is analysing the changes of the input vector $\boldsymbol{u}$ and the parameter vector $\boldsymbol{p}$. This two changing types are combined in one section, because the distinction between inputs and parameters is more a philosophical consideration. Both are assumed to be known for a particular model to compute the dynamics and the output. In the very beginning of the thesis, there was no distinction, both would be covered in the continuous external variables $\boldsymbol{w} \in \mathbb{R}^q$ of the continuous communication space. Several examples are known, where parameters are redefined as inputs or vice versa.

The version of a parameter change is the change of the values of the vector. This can be modelled in the considered environment by a transition between two states. Assume a current location $l \in L$ with the corresponding dynamics $\mathcal{D}_l$ given by $(f, g, h) \in \mathcal{D}_l$ and the equations

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases}$$

Due to the change of a parameter, the parameter should be assigned to the location and this means $\boldsymbol{p}_l$ is the current parameter of the location $l \in L$. The state event action is now formulated via the transition from the dynamics in the location $l \in L$ to the dynamics in the location $k \in L$. The transition is defined by the IODHSA, but to have as well a more abstract mathematical formulation, the transition is defined via an operator.

Definition 5.4. Let $\mathcal{M} = \{\mathcal{M}_l\}_{l \in L}$ be the set of all Models in a dynamic hybrid system environment and $\mathcal{E}_l$ the corresponding set of equations according to a given dynamic $\mathcal{D}_l$ in the given location $l \in L$. The *parameter change operator*, or *parameter transition operator*, is defined by

$$\mathcal{T}_{\boldsymbol{p}} \colon \mathcal{M} \to \mathcal{M}, \ \mathcal{E}_l \mapsto \mathcal{E}_k \,.$$

The operator $\mathcal{T}_{\boldsymbol{p}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t), \end{cases}$$

where $\boldsymbol{p}_l \in \mathbb{R}^{r_l}$ and $\boldsymbol{p}_k \in \mathbb{R}^{r_k}$.

In the given definition, only the parameter changes in the transition, which, of course, influences the dynamics.

Remark 5.5. Some remarks with respect to the involved symbolic:

(a) There is a difference between $\mathcal{M}$ and $\mathcal{M}_l$ for a certain $l \in L$. $\mathcal{M}$ is the aggregation of all models $\mathcal{M}_l$ which are used in a dynamic hybrid system description.

(b) The difference between the model description $\mathcal{M}_l$ and the continuous dynamics $\mathcal{D}_l$ is, that $\mathcal{M}$ includes the whole environment of a IODHSA, while $\mathcal{D}_l$ consists only of the corresponding equations. For a better overview of the addressed state event actions, the aggregation in a model description for the corresponding state is useful.

(c) There is as well a difference between $\mathcal{D}_l$ and $\mathcal{E}_l$. The dynamics is covering the collection of functions assigned to a location and meantime the equations as well cover the assigned parameters and inputs.

(d) In the case of the parameter changing state event, the change that could be noticed was, that in the equations of the particular dynamics, another parameter is inserted.

The change of the input can be translated one-to-one. The transition is again defined via an operator.

Definition 5.6. Assume the settings of Definition 5.4. The *input change operator*, or *input transition operator*, is defined by

$$\mathcal{T}_{\boldsymbol{u}}\colon \mathcal{M} \to \mathcal{M},\ \mathcal{E}_l \mapsto \mathcal{E}_k\,.$$

The operator $\mathcal{T}_{\boldsymbol{u}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t), \end{cases}$$

where $\boldsymbol{u}_l \in \mathbb{R}^{m_l}$ and $\boldsymbol{u}_k \in R^{m_k}$.

In both situations of transitions, the dynamics is the same. This means, the corresponding dynamics is only interrupted by changing values in the equations. For exemplification the following example is specified.

Example 5.7. Consider the differential equation system, proposed in [30], specified by

$$\begin{aligned} \dot{y}_1 &= -c_1 y_1 + c_1 y_2 + c_1 c_2, \\ \dot{y}_2 &= -c_3 y_2 + c_3 c_4, \end{aligned}$$

with $c_i \in \mathbb{R}$ for $i = 1, 2, 3, 4$ and scalar functions y_j for $j = 1, 2$. The parameters c_1 and c_3 are always constant, c_2 and c_4 are chosen according to the following definition of the following states:

(a) The parameters are fixed by $c_2 = \frac{4}{10}$ and $c_4 = \frac{11}{2}$ in this state, which is as well the initial state for the simulation. The initial conditions are given by $y_1(0) = \frac{21}{5}$ and $y_2(0) = \frac{3}{10}$. The model remains in this state as long as $y_1 < \frac{29}{5}$.

(b) In this state the parameters are given by $c_2 = -\frac{3}{10}$ and $c_4 = \frac{273}{100}$. The model remains in this state as long as $y_1 > \frac{5}{2}$.

This example fulfills exactly the required environment. By $y_1 < \frac{29}{5}$ and $y_1 > \frac{5}{2}$ the location invariants of both states are given as well as the guards and the parameter vector $\boldsymbol{p} = (c_1, c_2, c_3, c_4)^T$ is for both states given as described before. In this particular example, even the model structure is comfortable, due to a simple linear structure

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -c_1 & c_1 \\ 0 & -c_3 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} c_1c_2 \\ c_3c_4 \end{pmatrix}.$$

5.1.2 Change of the State Vector

This section is dedicated to the case of the state event which is changing the state vector by itself. This means, not the dynamics will be changed, but the value of the state vector. In the given environment, this change is done via the jump mapping. It is exactly the required transition for this kind of state event action. For unique notation, again an operator $\mathcal{T}_{\boldsymbol{x}}$ is defined, which updates the values of a state vector. In this situation, the value is changed, this means not, that a transition to another model description is necessary. In the environment of a dynamic hybrid system automaton, this corresponds to the transition in the same state. The jump mapping executes the update of the state vector values and the dynamic is continuing in the same location.

Definition 5.8. Assume the settings of Definition 5.4. The *state value change operator*, or *state value transition operator*, is defined by

$$\mathcal{T}_{\boldsymbol{x}} \colon \mathcal{M}_l \to \mathcal{M}_l, \ \boldsymbol{x} \mapsto J_{l,l}(\boldsymbol{x}),$$

where $J_{l,l}$ is a jump transition map, whereas the index describes the transition to the same location.

For illustration, an example introduced in the beginning of the thesis will be considered again.

Example 5.9. Reconsider subsection 2.3.2. The given example is the bouncing ball, with the given model description and the hybrid automaton illustrated in Figure 2.12.
The jump mapping was given via

$$\text{Jump:} \quad x_2 := -cx_2.$$

In the introduced environment, the equivalent formulation is done via

$$\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x} = J_{l,l}(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ -cx_2 \end{pmatrix}.$$

To understand this mapping in the setup of state events in the right way, once more the correct interpretation of the jump mapping is given above. If t^* denotes the event time, the equation above has to be interpreted as

$$\boldsymbol{x}(t^*_+) = J_{l,l}(\boldsymbol{x}(t^*_-)).$$

Beside the state jump inside a location, also a state jump to another location is possible. According to the environment, in fact it is only oriented to the (autonomous) guard and jump sets. This fact is covered by the operator formalism, because the operating state jump is executing the jump to another location. The state value change operator in this case is defined as

$$\mathcal{T}_{\boldsymbol{x}}\colon\ \mathcal{M} \to \mathcal{M},\ \boldsymbol{x} \mapsto J_{l,k}(\boldsymbol{x}), \tag{5.2}$$

for the locations $l, k \in L$.

5.1.3 Composition of Value Changes

The introduced value oriented state event changes are formulated via operators. This formalism, in the given form, allows to consider composed event actions, in form of composition of the defined operators. It is useful to include this in the formalism in order to offer the possibility of more complex state event actions.

Definition 5.10 (Composition of Value Transition Operators)**.** Assume the settings of Definition 5.4. The following compositions are defined:

(a) $\mathcal{T}_{\boldsymbol{p},\boldsymbol{u}} := \mathcal{T}_{\boldsymbol{p}} \circ \mathcal{T}_{\boldsymbol{u}} = \mathcal{T}_{\boldsymbol{u}} \circ \mathcal{T}_{\boldsymbol{p}}$. The operator $\mathcal{T}_{\boldsymbol{p},\boldsymbol{u}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t), \end{cases}$$

where $\boldsymbol{p}_l \in \mathbb{R}^{r_l}$, $\boldsymbol{p}_k \in \mathbb{R}^{r_k}$ and $\boldsymbol{u}_l \in \mathbb{R}^{m_l}$, $\boldsymbol{u}_k \in R^{m_k}$.

(b) $\mathcal{T}_{\boldsymbol{p},\boldsymbol{x}} := \mathcal{T}_{\boldsymbol{p}} \circ \mathcal{T}_{\boldsymbol{x}} = \mathcal{T}_{\boldsymbol{x}} \circ \mathcal{T}_{\boldsymbol{p}}$. The operator $\mathcal{T}_{\boldsymbol{p},\boldsymbol{x}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_l, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t), \\ g(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}_k, t), \end{cases}$$

where $\boldsymbol{p}_l \in \mathbb{R}^{r_l}$ and $\boldsymbol{p}_k \in \mathbb{R}^{r_k}$.

(c) $\mathcal{T}_{\boldsymbol{u},\boldsymbol{x}} := \mathcal{T}_{\boldsymbol{u}} \circ \mathcal{T}_{\boldsymbol{x}} = \mathcal{T}_{\boldsymbol{x}} \circ \mathcal{T}_{\boldsymbol{u}}$. The operator $\mathcal{T}_{\boldsymbol{u},\boldsymbol{x}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\mathcal{T}_{\boldsymbol{x}}\,\boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}, t), \end{cases}$$

where $\boldsymbol{u}_l \in \mathbb{R}^{m_l}$ and $\boldsymbol{u}_k \in \mathbb{R}^{m_k}$.

(d) $\mathcal{T}_{\boldsymbol{p},\boldsymbol{u},\boldsymbol{x}} := \mathcal{T}_{\boldsymbol{p},\boldsymbol{u}} \circ \mathcal{T}_{\boldsymbol{x}} = \mathcal{T}_{\boldsymbol{x}} \circ \mathcal{T}_{\boldsymbol{p},\boldsymbol{u}}$. The operator $\mathcal{T}_{\boldsymbol{p},\boldsymbol{u},\boldsymbol{x}}$ s the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t), \\ g(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}_l, \boldsymbol{z}, \boldsymbol{p}_l, t) \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\mathcal{T}_{\boldsymbol{x}}\, \boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t), \\ g(\mathcal{T}_{\boldsymbol{x}}\, \boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\mathcal{T}_{\boldsymbol{x}}\, \boldsymbol{x}, \boldsymbol{u}_k, \boldsymbol{z}, \boldsymbol{p}_k, t), \end{cases}$$

where $\boldsymbol{p}_l \in \mathbb{R}^{r_l}$, $\boldsymbol{p}_k \in \mathbb{R}^{r_k}$ and $\boldsymbol{u}_l \in \mathbb{R}^{m_l}$, $\boldsymbol{u}_k \in R^{m_k}$.

5.2 State Event Structural Changes

The second class of state event actions change the dynamics in the transition. Mainly the equations are addressed in the change of the state event actions, whereby in the first section the dynamics in the different locations are the same.

5.2.1 Change of the Derivative State Vector

Starting in a certain location $l \in L$ the continuous dynamics $\mathcal{D}_l$ together with the corresponding input and parameter creates a set of equation $\mathcal{E}_l$ which is given as

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases}$$

The state event action is oriented to the change of the derivative vector, i.e. the vector field f_l will be changed. As well for the structural changes, operators will be defined for executing the state event action.

Definition 5.11. Let $\mathcal{M} = \{\mathcal{M}_l\}_{l \in L}$ be the set of all models in a dynamic hybrid system environment and $\mathcal{E}_l$ the corresponding set of equations according to a given dynamic $\mathcal{D}_l$ in the given location $l \in L$. The *derivative state vector change operator*, or *derivative state vector transition operator*, is defined by

$$\mathcal{T}_{\dot{\boldsymbol{x}}} \colon \mathcal{M} \to \mathcal{M}, \; \mathcal{D}_l \mapsto \mathcal{D}_k \,.$$

The operator $\mathcal{T}_{\dot{\boldsymbol{x}}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases}$$

In this class of change characterisation the variable–structure systems of section 3.2 are included. Of course these systems are specialisations, because no algebraic equation is present, but the structure is comparable. One possible change is from a linear model to a nonlinear description, as it is in technical applications partly necessary.

In this characterisation, the issue regarding unique transitions can be addressed. Especially the parameter change and the state derivative vector change has some similarities. Reconsider Example 5.7. In this example the definition of the hybrid model was given via a parameter change. In the same way it can be as well defined via a derivative vector change, due to the presents of two different linear model structures. If the modelling is done via the one approach or the other it can not be graded objectivly. This means Example 5.7 could be also an example for a derivative vector change. This statement emphasises the statement that the parameter vector is also influencing the continuous dynamic of a certain location.

5.2.2 Change of the Algebraic Equation

The dynamic of each location is given by the differential part represented by the function f_l and the algebraic part, which interacts with the state vector by the algebraic equation g_l. A transition from one location to another allows to change this algebraic equation, which can be used for the characterisation of a corresponding change definition.

Definition 5.12. Assume the settings of Definition 5.11. The *algebraic equation change operator*, or *algebraic equation transition operator*, is defined by

$$\mathcal{T}_g \colon \mathcal{M} \to \mathcal{M},\ \mathcal{D}_l \mapsto \mathcal{D}_k\,.$$

The operator $\mathcal{T}_g$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g_l(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t), \\ g_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{p}, t). \end{cases}$$

Remark 5.13. The change of the algebraic equation g is similar to the change of the derivative vector, which is performed by changing the function f. This fact raises the question; why is not designed a change of the algebraic vector? This change is possible to define, but the role of the algebraic vector is the communication in the DAE system with the state vector. If a change of a certain number of a algebraic vector is necessary, these components can be added to the state vector and also performed there. In case of proper algebraic equations, in particular for DAE systems with differential index one, this model modification is possible.

5.2.3 Change of the Output Vector

The last component of the continuous dynamics is the output vector, which is defined via the output function h_l in a certain location $l \in L$. The change is performed as for the other changes of functions. Analogue is the definition of the corresponding operator.

Definition 5.14. Assume the settings of Definition 5.11. The *output vector change operator*, or *output vector transition operator*, is defined by

$$\mathcal{T}_{\boldsymbol{y}}\colon\ \mathcal{M} \to \mathcal{M},\ \mathcal{D}_l \mapsto \mathcal{D}_k\,.$$

The operator $\mathcal{T}_{\boldsymbol{y}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases}$$

5.2.4 Composition of Structural Changes

Identically to the composition of the value oriented changes, as well the structural change operators can be composed.

Definition 5.15 (Composition of Structural Transition Operators)**.** Assume the settings of Definition 5.11. The following compositions are defined:

(a) $\mathcal{T}_{\dot{\boldsymbol{x}},g} := \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_g = \mathcal{T}_g \circ \mathcal{T}_{\dot{\boldsymbol{x}}}$. The operator $\mathcal{T}_{\dot{\boldsymbol{x}},g}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases}$$

(b) $\mathcal{T}_{\dot{\boldsymbol{x}},\boldsymbol{y}} := \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_{\boldsymbol{y}} = \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_{\dot{\boldsymbol{x}}}$. The operator $\mathcal{T}_{\dot{\boldsymbol{x}},\boldsymbol{y}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases}$$

(c) $\mathcal{T}_{g,\boldsymbol{y}} := \mathcal{T}_g \circ \mathcal{T}_{\boldsymbol{y}} = \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_g$. The operator $\mathcal{T}_{g,\boldsymbol{y}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases}$$

(d) $\mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} := \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_g \circ \mathcal{T}_{\boldsymbol{y}}$. The operator $\mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}}$ executes the transition

$$\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}} = f_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases} \longmapsto \quad \mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}} = f_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t), \\ g_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t) = \boldsymbol{0}, \\ \boldsymbol{y} = h_k(\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{p},t). \end{cases}$$

Theorem 5.16. With the operator specifications of Definition 5.15 the following is valid.

(a) The composition of structural change operators is associative.

(b) The operator $\mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}}$ satisfies

$$
\begin{array}{lll}
\text{(a)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_{g}, & \text{(c)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{g} \circ \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_{\dot{\boldsymbol{x}}}, & \text{(e)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_{g}, \\
\text{(b)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{g} \circ \mathcal{T}_{\dot{\boldsymbol{x}}} \circ \mathcal{T}_{\boldsymbol{y}}, & \text{(d)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{\boldsymbol{y}} \circ \mathcal{T}_{g} \circ \mathcal{T}_{\dot{\boldsymbol{x}}}, & \text{(f)}\ \mathcal{T}_{\dot{\boldsymbol{x}},g,\boldsymbol{y}} = \mathcal{T}_{\dot{\boldsymbol{x}},g} \circ \mathcal{T}_{\boldsymbol{y}}
\end{array}
$$

and similar permuted compositions.

(c) The composition of structural change operators and value change operators is compatible.

5.3 State Event Model Change

The last state event action which has to be performed in the given formalism is the model change. This kind of change is the most drastic change of a certain model description in order which kind of changes take place during a transition from one location to the other. In this case, the maximum of the provided mathematical environment is used even though, the state space can be changed in this kind of state event action.

Definition 5.17. Let $\mathcal{M} = \{\mathcal{M}_l\}_{l\in L}$ be the set of all models in a dynamic hybrid system environment, $\mathcal{E}_l$ the corresponding set of equations according to a given dynamic $\mathcal{D}_l$ in the current location $l \in L$. The *model change operator*, or *model transition operator*, is defined by

$$\mathcal{T}_{\mathcal{M}} \colon \mathcal{M} \to \mathcal{M},\ \mathcal{M}_l \mapsto \mathcal{M}_k\,.$$

The operator $\mathcal{T}_{\mathcal{M}}$ executes in particular the transition

$$
\mathcal{E}_l \begin{cases} \dot{\boldsymbol{x}}_l = f_l(\boldsymbol{x}_l, \boldsymbol{u}_l, \boldsymbol{z}_l, \boldsymbol{p}_l, t), \\ g_l(\boldsymbol{x}_l, \boldsymbol{u}_l, \boldsymbol{z}_l, \boldsymbol{p}_l, t) = \boldsymbol{0}, \\ \boldsymbol{y}_l = h(\boldsymbol{x}_l, \boldsymbol{u}_l, \boldsymbol{z}_l, \boldsymbol{p}_l, t). \end{cases}
\longmapsto
\mathcal{E}_k \begin{cases} \dot{\boldsymbol{x}}_k = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k, t), \\ g_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k, t) = \boldsymbol{0}, \\ \boldsymbol{y}_k = h_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k, t). \end{cases}
$$

Of course, this state event action is the most complex change. Any change operator defined before the model change is a specialisation of this model change operator. The idea is, that this model change operator which in terms of changes, is the most expensive one, is used only in situations where the other operators are not sufficient. The main performance of the model change is the change of the state space by itself. This can not be performed with one of the operators before. Beside this restriction, the lower level changes can be implemented by composition of the other operators.

Remark 5.18. The term *model change* addresses the model of the continuous dynamics, not the discrete dynamics. These is given alone by the definition of the environment of the automaton. But, aim of the thesis is the mathematical characterisation of the changes in the dynamic hybrid models, the discrete environment is considered as the administration of the changes.

The model change state event action is possible due to the appropriate mathematical environment provided through the CGIODHSA, Definition 5.2. Each component of the hybrid state space provides in general, a continuous state space with different dimension. This last event action made it necessary to go so far in the generalisation of the mathematical environment.

5.4 Case Study: Oscillating Circuit with Diode

This section can be considered as the continuation of the subsection 2.3.3. In this subsection the oscillating circuit was combined with a switch. The switch was assumed to be externally controlled. This section will continue this example with the change of an internally controlled switch, represented by a diode. This case study will illustrate some particular transition operators. Furthermore, different diode models will be introduced, which leads to different model descriptions and different transitions.

The supposed circuit for the case study in this section is depicted in Figure 5.3

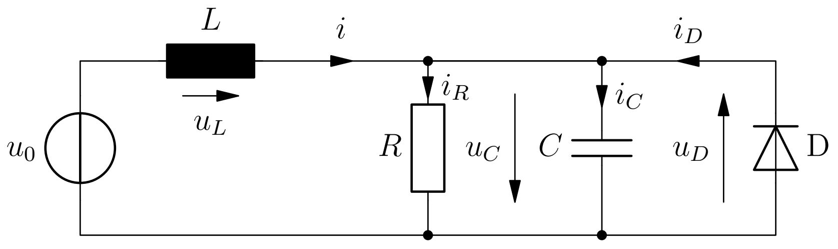


Figure 5.3: Example of an Electrical Circuit which includes a Diode.

In the following subsections different diode models and, as a consequence, different dynamic hybrid systems for the whole circuit will be discussed.

5.4.1 Ideal Switch Model

The first applied model for the diode is the model which is acting like a simple switch. The model equation for the voltage u_D and current i_D of the diode is given by the equation

$$u_D i_D = 0, \qquad u_D \leq 0, \qquad i_D \geq 0. \tag{5.3}$$

The corresponding characteristic is illustrated in Figure 5.4.

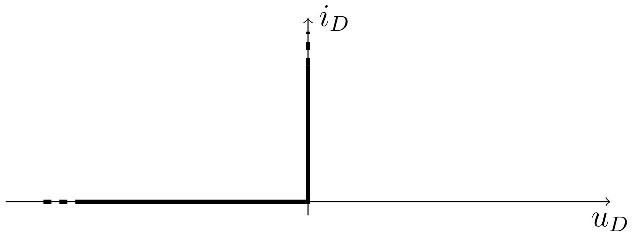


Figure 5.4: The Voltage–Current Characteristics of a Diode Model related to an Ideal Switch.

The benefit of this model is that it can be interpreted as an ideal switch, as used in the example from subsection 2.3.3, which is controlled by the voltage and current inside the circuit. The two operation modes of the diode distinguish the two locations of the dynamic hybrid system and can be separated from each other by the corresponding value of u_D or i_D. Furthermore, the location invariants of these locations can be derived, as listed in Table 5.1.

Location	Characterisation	Location Invariants
l_0	$i_D = 0$	$u_C = -u_D \geq 0$
l_1	$u_D = 0$	$i = -i_D \leq 0$

Table 5.1: Characterisation of Locations and their Location Invariants for the Oscillating Circuit with an Ideal Switch Diode Model.

In principle, this setup of the example is the system formulation from subsection 2.3.3. The switching process changes not to an autonomous switching process, on basis of the voltage and current distribution, and not longer via an input variable. To give an overview about the dynamic hybrid system, the description is summarised by

$$A = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ \cdot\frac{1}{L} & 0 \end{pmatrix} \qquad \text{and} \qquad B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

and the illustration of the hybrid automaton given in Figure 5.5. Jumps are not illustrated, because the jump mapping is the identity.

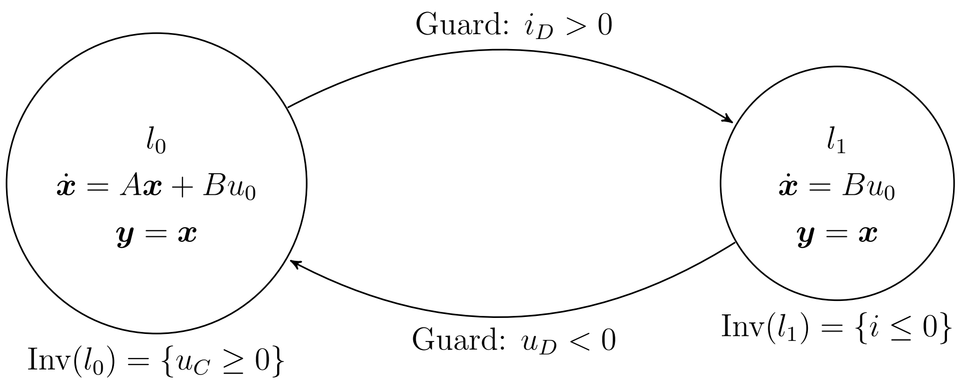

Guard: $i_D > 0$
l_0
$\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu_0$
$\boldsymbol{y} = \boldsymbol{x}$
l_1
$\dot{\boldsymbol{x}} = Bu_0$
$\boldsymbol{y} = \boldsymbol{x}$
Guard: $u_D < 0$
$\text{Inv}(l_0) = \{u_C \geq 0\}$
$\text{Inv}(l_1) = \{i \leq 0\}$

Figure 5.5: Hybrid Automaton of the Oscillating Circuit with an Ideal Switch as a Diode Model.

The present case is one of the state event transitions which are not unique. In principle the entries of the system matrix

$$A = \begin{pmatrix} a_{11} & a_{22} \\ a_{21} & a_{22} \end{pmatrix}$$

of the state space description can be assumed to be the parameter vector

$$\boldsymbol{p} = (a_{11}, a_{22}, a_{21}, a_{22})^T.$$

In this case, the state event can be performed as a parameter change with the corresponding transition operator

$$\mathcal{T}_{\boldsymbol{p}}\colon\ \mathcal{M} \to \mathcal{M},\ \dot{\boldsymbol{x}} = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} \boldsymbol{u} \longmapsto \dot{\boldsymbol{x}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} \boldsymbol{u}, \tag{5.4}$$

where $\boldsymbol{u} = u_0$. This approach is mathematically correct, but from an engineering perspective not useful. For engineering applications, the values of involved components, are assumed to be the parameter. In this case, the parameter vector is given by

$$\boldsymbol{p} = (R, L, C)^T \tag{5.5}$$

and the transition is covered by a derivative vector change with the associated operator

$$\mathcal{T}_{\dot{\boldsymbol{x}}}\colon \mathcal{M} \to \mathcal{M},\ \dot{\boldsymbol{x}} = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} \boldsymbol{u} \longmapsto \dot{\boldsymbol{x}} = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} \boldsymbol{u}. \tag{5.6}$$

This operator executes exactly the same transition as above, but with the parameter vector from equation (5.5). It is not useful to try to define always parameter state events, because the parameter are partially used also for other duties, like parameter studies, sensitivity analysis, etc. The choice of the parameter should not be influenced by the transition which has to be modeled and the parameter vector has to be chosen according to the modelling problem and duties.

5.4.2 Bent Characteristic Model

In order to bridge to a model for the diode, which is standard, this subsection will enhance the diode model from the last subsection and show the influences in the dynamic hybrid model of the circuit.

The first model adaption is the fact that diodes have normally a so-called breakthrough voltage, larger than zero. This means the diode characteristics will adjust, according to Figure 5.6.

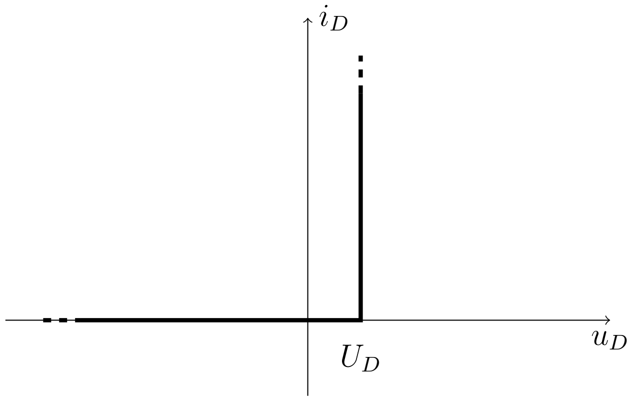


Figure 5.6: The Voltage–Current Characteristics of a Diode Model with Offset.

The equation, which describes this diode model with offset, is more complicated than equation (5.3) and is given through

$$u_D i_D = \begin{cases} 0, & \text{for } u_D \leq U_D, \\ sU_D, & \text{for } u_D = U_D, \end{cases} \tag{5.7}$$

with $s, U_D \in \mathbb{R}^+$. The corresponding dynamic hybrid system changes as well as the guards and location invariants have to be modified according to the inequality $u_D \leq U_D$. The model description for $u_D \leq U_D$ is the same, for $i \geq 0$ it has to be adapted according several restrictions. First, the diode model fixes

$$u_C(t) = -U_D,$$

hence

$$\dot{u}_C(t) = 0.$$

Furthermore, it is valid

$$i(t) = -i_D - \frac{U_D}{R}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t} i(t) = \frac{1}{L}(u_0 + U_D).$$

It is obvious that the model structure changes completely. A further modification addresses the slope of the line for $i_D \geq 0$. For a more realistic model, the slope has to be a finite real positive number, represented by a diode resistance R_D. The adapted characteristic is sketched in Figure 5.7.

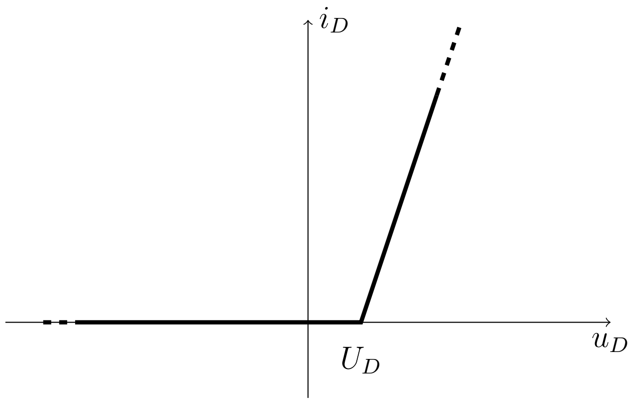


Figure 5.7: The Voltage–Current Characteristics of a Diode Model with Offset.

The related model equation is different compared to the previous model equations. In the current case, the characteristics can be formulated as a functional relation, separated in two cases, accordingly

$$i_D = \begin{cases} 0, & \text{for } u_D \leq U_D, \\ R_D(u_D - U_D), & \text{for } u_D > U_D, \end{cases} \tag{5.8}$$

Also this model adaption results in a more different model description for the location $u_D \leq U_D$, but the discrete dynamics do not change. For both specialisations of the diode model, a structural change operator is necessary. The aim of this subsection is to bridge to an interesting diode model, where the structural changes are different to the introduced ones.

5.4.3 Shockley–Equation Model

The final subsection is dedicated to a diode model, which results in an interesting dynamic hybrid model for the whole circuit. The purpose is to model the location for $u_D > 0$ via a differentiable function. The mathematical description is given by the equation

$$i_D = \begin{cases} 0, & \text{for } u_D \leq 0, \\ I_S(\mathrm{e}^{\frac{u_D}{U_T}} - 1), & \text{for } u_D > 0, \end{cases} \tag{5.9}$$

with $U_T, I_S \in \mathbb{R}^+$, corresponding to the characteristics given in Figure 5.8.

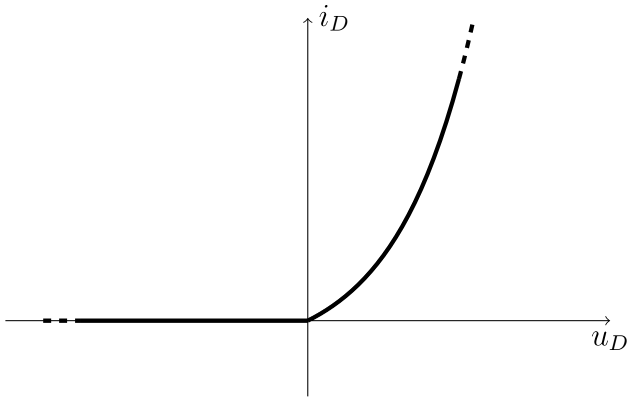


Figure 5.8: The Voltage–Current Characteristics of a Shockley–Equation Diode Model.

This changes the model structure for $u_D > 0$ completely. The basic equations are given by

$$i(t) = i_R(t) + i_C(t) - i_D, \qquad u_0(t) = u_L(t) + u_C(t)$$

and

$$u_C(t) + u_D = 0.$$

According to the component relations, this results in the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{pmatrix}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t) + \begin{pmatrix} \frac{1}{C} \\ 0 \end{pmatrix} i_D$$

and after inserting the Shockley–equation in

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} = \begin{pmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{pmatrix}\begin{pmatrix} u_C(t) \\ i(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix} u_0(t) + \begin{pmatrix} \frac{1}{C} \\ 0 \end{pmatrix} I_S\left(\mathrm{e}^{\frac{u_D}{U_T}} - 1\right) \tag{5.10}$$

Furthermore, the algebraic equation

$$u_c(t) + u_D = 0 \tag{5.11}$$

relates the differential equation with the state vector $\boldsymbol{x} = (u_C, i)^T$ and the algebraic vector $\boldsymbol{z} = u_D$.

In this setup, the transition is done via an algebraic equation change $\mathcal{T}_g$ which is special in this case because in the other location, the algebraic equation is not existent – represented by the zero function. Also the differential equation changes because in the location, analysed above, appears the algebraic variable. This means as well the transition operator $\mathcal{T}_{\dot{\boldsymbol{x}}}$ is used and in total, the transition $\mathcal{T}_{g,\dot{\boldsymbol{x}}}$ describes the state event action.

Chapter 6

Multi–Method Integration

The general aim of this thesis is to formulate a mathematical framework for dynamic hybrid models. In the previous chapters, the mathematical tools were defined and refined up to a certain standard, which fits to the given problems of system simulation. For system simulation, the main focus was the relation between the model $\mathcal{M}$ of a dynamic hybrid system and the particular subcomponents $\mathcal{M}_k$. The term submodel is here not appropriate, because the structures $\mathcal{M}_k$ are not all active for all time, they are frequented sequently, excluding the case of timeless locations. These are allowed to exist parallelly but this is not considered as the normal case.

In the previous chapter, finally the developed mathematical environment is used to formalise the transition operators for state event actions. For this formalism, the transition of the continuous dynamics is in the front. As illustrated as well in the case study of the past chapter, the transition between particular subcomponents involve the particular continuous dynamics $\mathcal{D}_l$, the corresponding equations $\mathcal{E}_l$ including the parameter vector $\boldsymbol{p}_l$ of the location $l \in L$. Via the hybrid state space X_l as well the discrete dynamics become important. Furthermore, the case study shows, that the transitions between the particular locations are covering major model changes, which are all covered by a generalised model environment $\mathcal{M}$ and their subcomponents $\mathcal{M}_k$, illustrated in Figure 6.1.

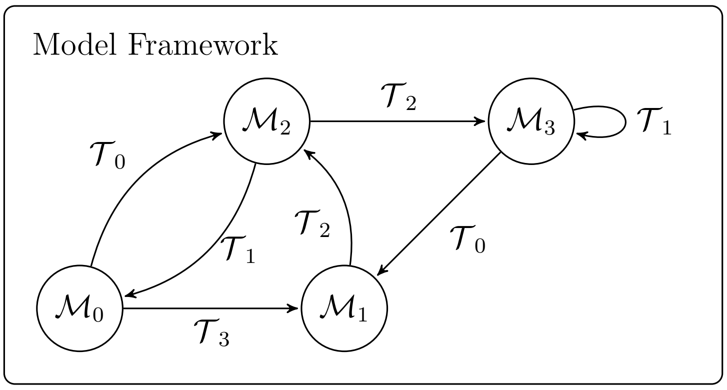


Figure 6.1: Illustration of a State Event Model with State Event Actions performed by universal Transition Operators $\mathcal{T}_i$.

This structure can be specialised in a certain way. The given environment assumes a uniform model structure in each location. Even in examples before, the model characterisations for each location, indeed, fit to the dynamic description, but the special cases could not be respected in a more simple model structure. Moreover, some situations maybe need other model descriptions, which are not covered by DAE models and could fit more to the situation and results in less cost and complexity in the step of implementation of the model. Discrete model structures, for instance, are not covered in the given continuous dynamics which would simplify the implementation of the given model environment in the next step.

Of course, this chapter is oriented more in the application of the model structure in a particular simulation environment and is not oriented to the structural mathematical modelling approach followed in the previous chapters and sections. This approach is the attempt to bring different model methods together and merge in one environment. Furthermore, the approach is oriented on modelling approaches which are well known and applied in different fields of application, even in disciplines which are located on the boarder of technical and natural science fields.

For introducing the idea of expansion to other modelling techniques, beside DAE models, the chapter will focus on the following structure:

(a) *Modelling Aspects:* Differentiation between an abstract mathematical model, a model which includes a structural environment – proper for simulation – and a model ready for implementation in a particular model environment.

(b) *Interface Aspects:* To widen the range of continuous dynamics $\mathcal{D}_l$ in the particular location $l \in L$, the definition of an communication interface is necessary. This interface will guarantee the presents of the necessary information to fit in the discrete dynamics of the model $\mathcal{M}$.

(c) *Encapsulation Effects:* If the interfaces in (b) are provided, the continuous dynamics can differ in the sense of detailing. In this sense, models of different level of detail can be presented in different subcomponents $\mathcal{M}_k$ and connect particular properties to a whole model $\mathcal{M}$.

(d) *Categorisation and Model–Layer:* The different level of detail in (c) allows from a modelling point of view to categories different subcomponents and different modelling layers.

6.1 Modelling Aspects for Dynamic Hybrid Models

This thesis follows the modelling progress in the following structure:

(a) *Abstract Model:* The given modelling problem, either is given in an abstract mathematical formulation or in the context of the corresponding scientific field by convenient methods. In this phase no aspect of a later simulation is present.

(b) *Structural Model:* This stage describes a mathematical formulation, which is suitable for a procedural numerical handling in a simulation environment. The structure is given by the simulation time, thus a later processing is possible.

(c) *Simulation Model:* In this step, the model is adapted to available modelling methods which are available in a certain simulation environment.

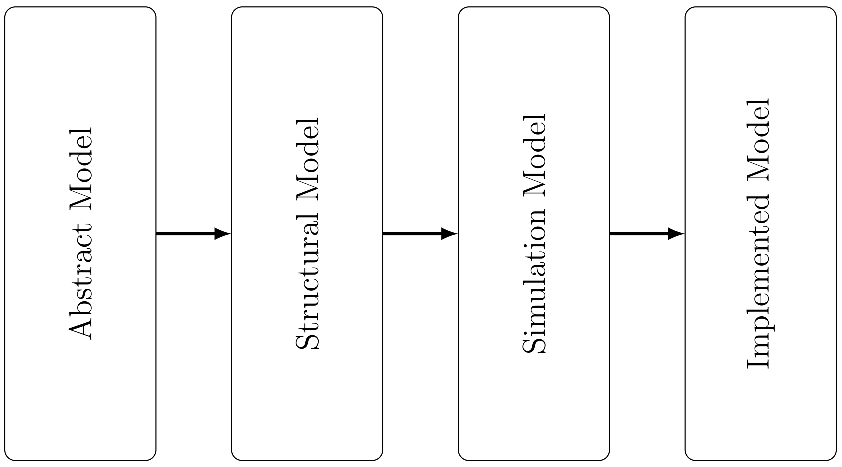


Figure 6.2: Modelling Progress from the abstract Model up to the implemented Model.

(d) *Implemented Model:* The final stage is the implemented model, which contains as well numerical considerations and settings.

The structured modelling process is illustrated in Figure 6.2.

The presented mathematical framework of dynamic hybrid models is located in the structural model phase of the modelling progress. The automaton representation shows an inherent time progression through a sequence of active subcomponents. The subcomponents are not related to the simulation abilities of a certain environment and as well not oriented to numerical considerations of the implementation.

The following chapter will widen the range of continuous dynamics, in terms of including as well different modelling structures instead of DAE models for the subcomponents. The presented approach will be given on the structural model layer, but of course it concerns also some performance considerations in the numerical environment.

6.2 Extension of the Continuous Dynamics

A system dynamics on an abstract layer is defined in Definition 2.1. Furthermore, there is shown how this flow formalism can be translated to a differential equation, which characterises the dynamics of a system. This is the first indicator that, for modelling purposes, the inclusion of more than the introduced dynamics $\mathcal{D}$ is useful. This section will execute this extension.

6.2.1 Definition of Multi–Method Dynamic Hybrid Models

The model of the subcomponents $\mathcal{M}_l$ includes up to now the continuous dynamics $\mathcal{D}_l$ and the corresponding equations $\mathcal{E}_l$ which were influenced by the parameter vector $\boldsymbol{p}_l$. Moreover, the subcomponent was as well associated to the discrete dynamics given by the guard and jumps, respectively the corresponding autonomous and controlled jump sets and transition maps in the current location $l \in L$. In the next refinement of dynamic hybrid models, the discrete dynamics will be unmodified and as well the involved vectors of the continuous dynamics will be preserved.

Only the associated functions in $\mathcal{D}$ and as a consequence the corresponding equations $\mathcal{E}$ will be modified. The modification is more a generalisation, because the mathematical description has still to fulfill the requirements of a dynamic system given in Definition 2.1.

Definition 2.1 allows as well time discrete dynamic systems, which are interesting especially for simulation purposes. If the associated dynamics can be specified, the dynamic hybrid model $\mathcal{M}$ can consist of subcomponents $\mathcal{M}_l$ which are defined by a different continuous dynamic representing different modelling methods. Due to this aspect, the inclusion of a wider range of dynamics, the model $\mathcal{M}$, will be considered as one using multiple methods.

Definition 6.1. Let $\mathcal{M} = \{\mathcal{M}_l\}_{l\in L}$ be the set of all Models in a dynamic hybrid system environment.

(a) $\mathcal{D}_l$ specifies the *generalised continuous dynamics* and $\mathcal{E}_l$ the corresponding set of mathematical formulations of the generalised continuous dynamics in the given location $l \in L$ with the associated vectors $\boldsymbol{x}_l$, $\boldsymbol{u}_l$, $\boldsymbol{z}_l$ and $\boldsymbol{p}_l$.

(b) The given model in a dynamic hybrid system environment is called a *multi–method dynamic hybrid model.*

Example 6.2. Some examples of continuous dynamics for multi–method models:

(a) *Discrete dynamical systems.* In this case the differential calculus will be replaced by a sequence calculus. The differential dynamics is given by the equation

$$\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{z}_n, \boldsymbol{p}_n),$$

where $\boldsymbol{x}_n$,$\boldsymbol{u}_n$, $\boldsymbol{z}_n$ and $\boldsymbol{p}_n$ represents the vectors of sequences of the state, input, algebraic and parameter vector. Equivalent the vector of the output sequence is given by

$$\boldsymbol{y}_n = h(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{z}_n, \boldsymbol{p}_n).$$

(b) *Cellular Automata.* The continuous dynamics can as well be described by one dimensional cellular automata. This modelling technique is used for spatial discrete settings. In case of one spatial dimension, the automaton can be used directly and in case of more spatial dimensions, some accumulated values have to be used for the dynamic system output. A deeper view in cellular automata is provided in the publication of [51] and in the thesis [47].

(c) *Agent Based Structure.* This method describes a modelling approach, which starts from a cohort of individual agents who are interacting with each other according to defined local rules. This individual based approach can not be used directly as a continuous dynamics. It has to be adopted to the requirements of the mathematical environment or proper accumulated values have to be used for the connection to the discrete dynamics. Nevertheless, it is an appropriate method for modelling dynamic processes. A detailed introduction and the proof that, agent based models are equivalent, in a certain meaning, to differential equations is presented in [2].

In this aspect, the modelling approaches start to be relevant. As presented in the introduction of this thesis, chapter 1, the starting point of the presented form of dynamic hybrid models is given. The aim is a structural model, as it is named in this chapter. But, the modelling approach consider only DAE models in the particular locations. The background is simple and the presented framework comes from engineering and partly natural science subjects, where the characterisation of dynamic systems is usually done by differential equations. Algebraic equations are accepted because of the restrictions and dependencies in the dynamic systems and thus this approach accommodate the way of working in these fields.

From a modelling point of view, the approach is different. Not that the modelling method should be known and the problem has to be adjusted to the method, but the way around. The proposed concept, to enable more frameworks for the continuous dynamics of the defined structure of a dynamic hybrid system, meets this modelling approach.

Remark 6.3. The presented extension of valid continuous dynamics in the definition of a multi–method dynamic hybrid model mixes the different layers of the modelling queue, illustrated in Figure 6.2. In the setting, now it is possible to use time–continuous dynamics and time–discrete dynamics in the different locations. This enables as well, that a discretisation of a time–continuous model description can be considered as the dynamic in a certain location by itself. It is obvious, that concerns regarding implementation and simulation time in the modelling phase can force to a model description where this concerns are incorporated.

6.2.2 Interface Definition for Subcomponents

Basically, the continuous dynamics $\mathcal{D}_l$ in a certain location $l \in L$ has to fulfill a rather simple property. The active dynamics has to provide for the model description the required vectors over time. This allows the definition of an interface in that meaning, that each dynamic has to provide the required vector $\boldsymbol{x}_l$ and has to be able to process the vectors $\boldsymbol{u}_l$, $\boldsymbol{p}_l$. The vector $\boldsymbol{z}_l$ is a property of the location by itself and is not necessary to be provided. As well several time–continuous dynamics does not provide this vector, but if it is necessary for the model description in the location by itself or the location transition, the vector has to be provided.

The representation of a multi–method dynamic hybrid model is again an automaton. A particular multi–method dynamic hybrid model can involve the classical definition of a dynamic hybrid model together with a generalised dynamic. Figure 6.3 illustrates a certain automaton.

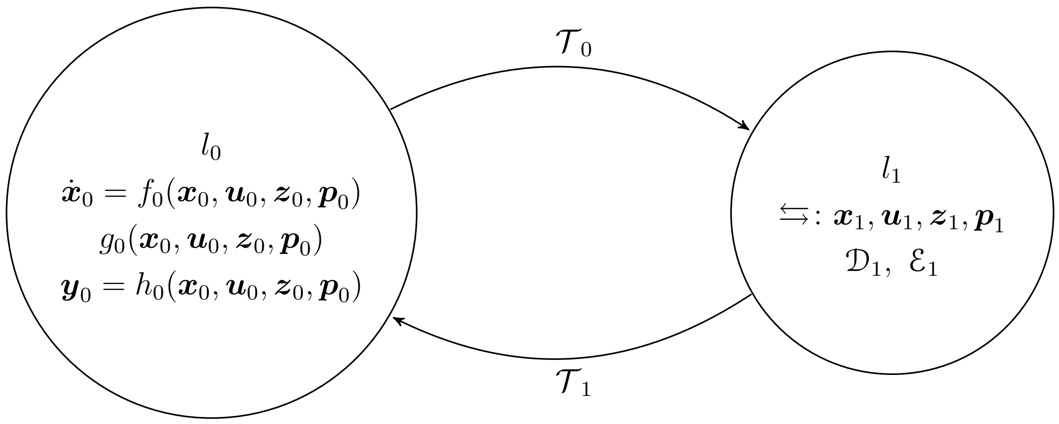

$\mathcal{T}_0$
l_0
$\dot{\boldsymbol{x}}_0 = f_0(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{z}_0, \boldsymbol{p}_0)$
$g_0(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{z}_0, \boldsymbol{p}_0)$
$\boldsymbol{y}_0 = h_0(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{z}_0, \boldsymbol{p}_0)$
l_1
$\leftrightarrows: \boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{z}_1, \boldsymbol{p}_1$
$\mathcal{D}_1,\ \mathcal{E}_1$
$\mathcal{T}_1$

Figure 6.3: Representation of a two Location Multi–Method Dynamic Hybrid Model.

The symbol $\leftrightarrows$, in Figure 6.3, indicate the *interface* for the location l_1, where $\boldsymbol{x}_1$ has to be provided, whereas $\boldsymbol{u}_1, \boldsymbol{p}_1$ has to be processed from the dynamics. If $\boldsymbol{z}_1$ is required, depends on the transition from ant to the location in the particular example.

6.2.3 Encapsulation Effects in Subcomponents

In the beginning of this section, the idea was introduced to use continuous dynamics for particular locations which fit to the mathematical framework and is useful from a modelling point of view. In the last subsection, the idea of an interface was introduced, which separates the continuous dynamics from the location against the rest of the automaton network. This interface definition has an interesting side effect. The location can be assumed in the first step as an black box, where $\boldsymbol{u}_l$, $\boldsymbol{p}_l$ are considered as the input and $\boldsymbol{x}_l$, $\boldsymbol{y}_l$ as the output. Beside, the state vector value after the jump, $\boldsymbol{x}_l^0$, has to be provided which concerns the interface from the last subsection. The black box approach allows that continuous dynamics of the location has the possibility to offer a wider range of values and mathematical environment, than it is required for the multi–method dynamic hybrid model – as illustrated in Figure 6.4. This effect is called *encapsulation*.

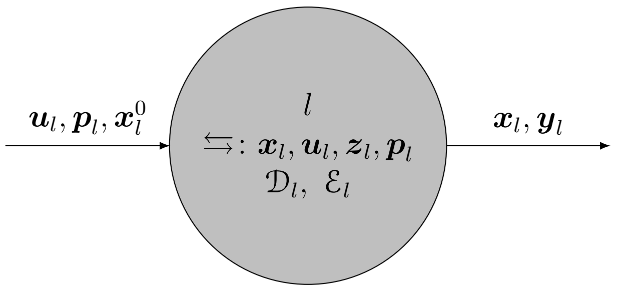

$\boldsymbol{u}_l, \boldsymbol{p}_l, \boldsymbol{x}_l^0$
l
$\leftrightarrows: \boldsymbol{x}_l, \boldsymbol{u}_l, \boldsymbol{z}_l, \boldsymbol{p}_l$
$\mathcal{D}_l,\ \mathcal{E}_l$
$\boldsymbol{x}_l, \boldsymbol{y}_l$

Figure 6.4: Black Box Modelling Approach for a certain Location of a Multi–Method Dynamic Hybrid Model.

6.3 Characterisation of the Multi–Method Environment

According to the purpose of the thesis, as well the extended environment has to be investigated regarding characterisation and state events. The aspect of the state events in the discrete dynamics is not changed and guards and jumps handle this. In the aspects of state event action, the point is different. The transitions are no longer homogeneous, in terms of operating on the same set of submodels $\mathcal{M}_k$ in $\mathcal{M}$, the continuous dynamics can change completely, even the mathematical way how to quote the model description. This influences the already defined state transition operators. Several are re-usable and others has to be adapted. The following subsections will discuss this aspects, furthermore the categorisation of the involved subcomponents and the possibility of composition.

6.3.1 State Event Characterisation

In chapter 5, Mathematical Characterisation of State Events change transition operators are defined to formalise and characterise continuous dynamic changes. Due to the extension of the concept of a continuous dynamics, the defined transition operators has to be reviewed according to the new dynamic definition. The transition operators are characterised in value change, structural change and model change operators. A transition in a multi–method model is by definition a structural change, due to the focus of the usage of several different continuous dynamics.

In the current setup, the change of particular dynamics to one of the same type is a special case. If the dynamics of the equal type are DAE models, the transition operators of chapter 5 are still valid. Also a transition of the continuous dynamics $\mathcal{D}$, which is not a DAE type, to the same type of dynamics is a special case, because of the transitions between similar mathematical structures. This transition within a continuous dynamics formalism can be formalised.

Definition 6.4. Assume a multi–method dynamic hybrid model $\mathcal{M}$, with multi–method subcomponents $\{\mathcal{M}_l\}_{l\in L}$. Each location is associated with a generalised continuous dynamics $\mathcal{D}_l$ and the corresponding set of mathematical formulation $\mathcal{E}_l$. Moreover, assume a set of locations $\{l_1,\ldots,l_p\}$ for whom the continuous dynamics are from equal type.

(a) The set $\{\mathcal{M}_{l_1},\ldots,\mathcal{M}_{l_p}\}$ is the collection of multi–method subcomponents with equal continuous dynamics and is denoted as $\mathcal{M}_e$.

(b) A transition operator, defined by

$$\mathcal{T}^{\circ}\colon\ \mathcal{M}_e \to \mathcal{M}_e$$

is called an *intra method transition operator*.

The link to the corresponding transition operators for DAE continuous dynamics is given by the following Corollary.

Corollary 6.5. The transition operators of chapter 5 are all intra method transition operators.

To complete the discussion about state event actions in multi–method dynamic hybrid models, also a transition between different continuous dynamics has to be considered.

Definition 6.6. Assume a multi–method dynamic hybrid model $\mathcal{M}$, with multi–method subcomponents $\{\mathcal{M}_l\}_{l\in L}$. Each location is associated with a generalised continuous dynamics $\mathcal{D}_l$ and the corresponding set of mathematical formulation $\mathcal{E}_l$. Moreover, assume two locations l, k with a different type of generalised continuous dynamics, $\mathcal{D}_l$ and $\mathcal{D}_k$. A transition operator, defined by

$$\mathcal{T}^{c}\colon\ \mathcal{M} \to \mathcal{M},\ \mathcal{D}_l \mapsto \mathcal{D}_k$$

is called an *inter method transition operator*.

6.3.2 Categorisation of Subcomponents

The definition of intra and inter method transition operators leads to the idea that within a multi–method dynamic hybrid model, subcomponents of equal type can be collected. These equal types refer to the equal types of continuous dynamics. A categorisation w.r.t. the continuous dynamics would result in a clustering of the model $\mathcal{M}$. This cluster would provide a certain topology in the subcomponents and it is clear, that inside a cluster component different types of transition operators acting than outside. The following definition formalise this concept.

Definition 6.7. Assume a multi–method dynamic hybrid model $\mathcal{M}$, with multi–method subcomponents $\{\mathcal{M}_l\}_{l\in L}$. Assume in a multi–method dynamic hybrid model n different types of continuous dynamics. Similar to Definition 6.4 (a), the subcomponents will be clustered in n collection of subcomponents where within one collection, continuous dynamics are from equal type. The set of the n collections of subcomponents $\{\mathcal{M}_{e_1}, \mathcal{M}_{e_1}, \ldots, \mathcal{M}_{e_n}\}$ is called a *cluster of multi–method subcomponents* with equal continuous dynamics and they satisfy

$$\mathcal{M} = \bigcup_{i=1}^{n} \mathcal{M}_{e_i} .$$

It is obvious, that inside one cluster component $\mathcal{M}_{e_i}$ an intra method transition operator executes the location change and between two cluster components $\mathcal{M}_{e_i}$ and $\mathcal{M}_{e_j}$ an inter method transition operator executes the location change. In this meaning the definition of the cluster provides a categorisation of subcomponents.

Chapter 7

Final Remarks and Summary

The thesis, on mathematical characterisation of state events in hybrid modelling, presentes the union of a mathematical framework for hybrid models with the aspects of state event modelling for system simulation. The main focus, of almost two-thirds of the thesis, is dedicated to the continuous dynamic description done by differential algebraic equation. The change of the model description, which is introduced in the state event procedure, is replaced by defining a mathematical foundation and defining the transition in this environment. Several chapters were dedicated in introducing the structure and verify the needs from a system point of view, but as well from a simulation environment point of view.

The aim of the thesis was the mathematical formulation of a structural model for a given setup. The thesis does not discuss the modelling process up to the stage of a hybrid model. As mentioned in the introduction, this modelling aspect is not a subject to be addressed. A major disadvantage of hybrid model in general is that, the model description is often given in certain formalism out of the subject area, or due to the aim of simulation in a model description which uses already the formalism of a certain simulation environment. This mixing of mathematical model and the model realisation in a certain simulation environment does not allow the objective analysis of hybrid model or state event changes. The presented mathematical environment offers the possibility to last exactly this.

Over a long distance, the structural modelling point is respected in the thesis. The last step to extend the theory to multi–method approaches leads to leaving the structural point of view. The motivation is to allow different model description for continuous dynamics in the location, to fulfill certain criteria regarding efficiency and optimisation in the simulation process. The reason to use a time–discretized version of a time–continuous description regards normally this aspects.

In the thesis, also for this multi–method version of a dynamic hybrid model, a characterisation of state transitions is given. But, the present characterisation is on an abstract level.
Further work can address this aspect and continue the description for several involved continuous dynamics.

Especially, including certain spatial models can be interesting. Simulation environments try to fulfill this idea. The combination of a simulator, which can handle the dynamical system specification and as well submodels spatial systems are included. Applied on the focus of this thesis, would mean that, for a certain scenario, as well spatial models are included.

One of the arguments for extension to multi–method aspects was, to include other modelling approaches. A certain class of modelling approach is the field of fuzzy systems and neuronal network which sometimes are covered by the term soft computing. The modelling methods are common and applied in several fields of applications. To include this modelling methods, the range has to extend to stochastic models. This is a wide field, because the stochastic aspect can influence the given model foundation in several aspects. Either inputs are assumed to be stochastic, or the discrete structure will allow stochastic transitions, etc. Several possible extensions are available in this field. Some aspects are addressed and covered in [40] but according to the aim of the presented thesis, several definitions and extensions of the structure are necessary.

Furthermore, studying the algebraic structure can be interesting. As shown in Definition 3.13 for hybrid automaton, the dynamic hybrid systems can be composed as well. In particular, the possibility to use multi–method dynamic hybrid models as models for other modelling processes or use a model of this type as one component in another multi–method dynamic hybrid model. For the moment, the current given structure fails because of the missing of a corporate description of the continuous dynamics for the whole multi–method dynamic hybrid model.

To summarise, the field of hybrid models is highly interesting, because the model description is very powerful and offers a variety of applications. Mathematical modelling and simulation enters in more and more disciplines and subjects. There are only few technical sciences of applied sciences which can work without simulation. The increasing computer power raises the interest to model more detailed processes and structures and as a result, the models are getting more complex. This surrounding is perfectly matching for the application of hybrid model in the future.

List of Figures

List of Tables

Bibliography

[1] A. Bemporad. Hybrid Toolbox - User's Guide, 2004. `url:` cse.lab.imtlucca.it, May 2015.

[2] M. Bicher. Agentenbasierte Modellbildung und Simulation auf Basis der Fokker–Planck–Gleichung. Master's thesis, Vienna University of Technology, 2013.

[3] A. Borshchev. Multi–method modelling. In *Proceedings of the 2013 Winter Simulation Conference*, pages 4089–4100, 2013.

[4] M. S. Branicky. *General hybrid dynamical systems: Modeling, analysis, and control*, pages 186–200. Springer, 1996.

[5] M. S. Branicky. *Handbook of Networked and Embedded Control Systems*, chapter Introduction to Hybrid Systems, pages 91–116. Birkhäuser Boston, 2005.

[6] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control. In *33rd IEEE Conf. Dec- Contr.*, pages 4228–4234, 1994.

[7] F. Breitenecker, H. Ecker, B. Heinzl, A. Körner, M. Rößler, and N. Popper. Change of Independent Variable for State Event Detection in System Simulation – Evaluation with ARGESIM Benchmarks. In *Proceedings of the 24th European Modelling & Simulation Symposium*, pages 531–536, 2012.

[8] F. Breitenecker, A. Körner, and N. Popper. About an Alternative Method of Numerical Iteration for State Event Finding and Handling in System Simulation of Hybrid Dynamical Systems. In *Proceedings of 2013 UKSim 15th International Conference on Computer Modelling and Simulation (UKSim)*, pages 390–395, 2013.

[9] M. L. Bujorianu and J. Lygeros. General stochastic hybrid systems: modelling and optimal control. In *43rd IEEE Conference on Decision and Control (Volume:2)*, pages 1872–1877, 2004.

[10] F. E. Cellier and E. Kofman. *Continuous System Simulation.* Springer, New York, 2006.

[11] J. A. Cid and R. L. Pouso. On first–order ordinary differential equations with nonnegative right-hand sides. *Nonlinear Analysis: Theory, Methods & Applications*, 52:1961–1977, 2002.

[12] H. Dai, X. Yue, J. Yuan, D. Xie, and S. Atluri. A comparison of classical Runge–Kutta and Henon's methods for capturing chaos and chaotic transients in an aeroelastic system with freeplay nonlinearity. *Nonlinear Dynamics*, 79:10.1007/s11071–015–1980–x, 2015.

[13] X. C. Ding. *Real–time optimal Control of Autonomous Switched Systems.* PhD thesis, Georgia Institute of Technology, 2009.

[14] O. El-Ghezawi and A. Zinober. Variable Structure Systems and System Zeros. Research Report No. 157, University of Sheffield, England, 1981.

[15] S. Engell, G. Frehse, and E. Schnieder, editors. *Modelling, Analysis and Design of Hybrid Systems*. Springer, Berlin Heidelberg, 2002.

[16] J. M. Esposito and V. Kumar. A state event detection algorithm for numerically simulating hybrid systems with model singularities. *ACM Trans. Model. Comput. Simul.*, 17(1), 2007.

[17] G. Ferrari-Trecate. Hybrid Identification Toolbox (HIT), 2005. `url:` sisdin.unipv.it/lab/, May 2015.

[18] P. A. Fishwick, editor. *Handbook of Dynamic System Modelling*. Chapmann & Hall/CRC, Taylor & Francis Group, Boca Raton, London, New York, 2007.

[19] E. Frazzoli, M. A. Dahleh, and E. Feron. *System Theory: Modeling, Analysis, and Control*, chapter A Hybrid Control Architecture for aggressive Maneuvering of autonomous aerial Vehicles, pages 325–244. Springer Science+Business Media New York, 2000.

[20] M. Glocker. *Diskret–kontinuierliche Optimalsteuerung: Modellierung, Numerik und Anwendung bei Mehrfahrzeugsystemen*. PhD thesis, Technische Universität Darmstadt, 2005.

[21] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems*. Princeton University Press, Princeton, 2012.

[22] S. Hara and H. Fujioka. A Hybrid State–Space Approach to Sampled–Data Feedback Control. *Linear Algebra and its Applications*, 205–206:675–712, 1994.

[23] W. P. M. H. Heemels, S. Weiland, and A. L. Juloski. Input–to–State Stability of Discontinuous Dynamical Systems with an Observer–Based Control Application. In *10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007. Proceedings*, pages 259–272, 2007.

[24] M. Henon. On the Numerical Computation of Poincare Maps. In *Physics 5D*, pages 412–414, 1982.

[25] T. A. Henzinger. *Verification of Digital and Hybrid Systems*, chapter The Theory of Hybrid Automata, pages 265–292. Springer Berlin Heidelberg, 2000.

[26] J. P. Hespanha. Hybrid and Switched Systems. Lecture Notes, University of California, 2005.

[27] H. Heuser. *Gewöhnliche Differentialgleichungen*. Vieweg+Teubner, Wiesbaden, 2008.

[28] D. Hinrichsen and A. J. Pritchard. *Mathematicsl System Theory I*. Springer, Berlin Heidelberg, 2005.

[29] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, New York, 1970.

[30] N. Houbak. Software Comparison 5. *Simulation News Europe*, 2:31, 1992.

[31] G. Karer, G. Music, I. Skrjanc, and B. Zupancic. Feedforward control of a class of hybrid systems using and inverse model. *Mathematics and Computers in Simulation*, 82:414–427, 2011.

[32] S. Kowaleski, M. Garavello, and et al. *Handbook of Hybrid Systems Control*, chapter Hybrid Automata, pages 57–86. Cambridge University Press, 2009.

[33] A. Körner. Analyse und Simulation dynamischer Systeme in Simulink mit Web–Interface. Master's thesis, Vienna University of Technology, 2012.

[34] A. Körner and F. Breitenecker. Approaches for State Event Handling by Simulation Algorithm and via Model Description. In *Tagungsband ASIM 2014 – 22. Symposium Simulationstechnik*, pages 219–224, 2014.

[35] A. Körner, B. Heinzl, M. Rößler, and et al. BCP - A Benchmarik for Hybrid Modelling and State Event Modelling. In *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation*, pages 1032–1042, 2010.

[36] D. Liberzon. *Switched Systems: Stability Analysis and Control Synthesis.* Birkhäuser Boston, Boston, 2003.

[37] D. Liberzon. *Switched systems*, pages 559–574. Birkhäuser Boston, 2005.

[38] J. Lygeros, C. Tomlin, and S. Sastry. *Hybrid Systems: Modeling, Analysis and Control.* EECS Instructional Support Group, University of California, Berkeley, 2008.

[39] F. Niewels and H. Kiendl. Design and Simulation of Structure–Variable Control Systems Based on Multi–Valued Lyapunov Functions. *Systems Analysis Modelling Simulation*, 42:1199–1217, 2002.

[40] B. Nixdorf. *Discrete–Event Modelling and Control of Hybrid Systems.* PhD thesis, Technische Universität Hamburg–Harburg, 2002.

[41] T. Park and P. I. Burton. State Event Location in Differential–Algebraic Models. *ACM Transactions on Modeling and Computer Simulation*, 6:137–165, 1996.

[42] B. Potocnik, G. Music, I. Skrjanc, and B. Zupancic. Model-based Predictive Control of Hybrid Systems: A Probabilistic Neural-network Approach to Real-time Control. *Journal of Intelligent and Robotic Systems*, 51:45–63, 2008.

[43] B. Potocnik, G. Music, and B. Zupancic. A New Technique for Translating Discrete Hybrid Automata into Piecewise Affine Systems. *Mathematical and Computer Modelling of Dynammical Systems*, 10:41–57, 2004.

[44] B. Potocnik, G. Music, and B. Zupancic. Model predictive control of discrete–time hybrid systems with discrete inputs. *ISA Transactions*, 44:199–211, 2005.

[45] B. Potocnik and B. Zupancic. Modelling of Hybrid Systems by MLD. *Simulation News Europe*, 10:16–17, 2000.

[46] A. Sabanovic. Variable Structure Systems with Sliding Modes in Motion Control – A Survey. *IEEE Transactions on Industrial Informatics*, 7:212–223, 2011.

[47] G. Schneckenreither. Developing mathematical formalisms for cellular automata in modelling and simulation. Master's thesis, Vienna University of Technology, 2014.

[48] The MathWorks Inc. MATLAB Documentation. www.mathworks.com/help/matlab/, May 2015.

[49] F. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and D. Mignone. *HYSDEL - User Manual*, 2002.

[50] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Systems.* Springer, London, 2000.

[51] S. Wolfram. Cellular automata as a model of comlexity. *Nature*, 311:419–424, 1984.

[52] X. Xu and P. J.Antsaklis. Optimal control of hybrid autonomous systems with state jumps. In *Proceedings of the 2003 American Control Conference*, pages 5191–5196, 2003.

[53] G. Zhai, X. Xu, H. Lin, and A. N. Michel. Analysis and design of switched normal systems. *Nonlinear Analysis*, 65:2248–2259, 2006.

Index

Curriculum Vitae

Andreas KÖRNER

22.03.1982, male
Austria, 1210 Vienna, Donaufelderstraße 57/2/8

Education and Academic Career

1996–2001	University Entrance Diploma at Technical Collage in Vienna
2001–2002	Military Service
2002–2009	Vienna University of Technology, BSc Electrical Engineering and MSc Telecommunication
2004–2012	Vienna University of Technology, Diploma Studies Technical Mathematics
2006–2009	Tutor at Institute for Analysis and Scientific Computing, Vienna University of Technology
2007-2009	Tutor at Institute of Communications and Radio Frequency Engineering, Vienna University of Technology
2011–2015	Doctoral Program in Technical Mathematics
Since 2009	Project Assistant at Institute for Analysis and Scientific Computing, Vienna University of Technology
April–May 2012	Visiting Researcher at University of Ljubljana, Fac. of Electrical Engineering, Laboratory of Modelling, Simulation and Control
April–June 2014	Guest Lecturer at University for Business and Technology, Pristina, Kosovo
April 2015	Guest Lecturer at University Tirana, Fac. of Economy, Tirana, Albania

Academic Activities

February 2009	Conference Assistant at 6th Vienna International Conference on Mathematical Modelling, 11th–13th February
February 2012	Member of National Organisation Committee at 7th Vienna International Conference on Mathematical Modelling, 14th–17th February
Since 2013	Coordinator of class schedule at Faculty for Mathematics, Vienna University of Technology
2013	Member in College LehreN - Encouragement for Teachers in Engineering aided by German Beneficence Consortium
February 2015	Chair of the National Organisation Committee at 8th Vienna International Conference on Mathematical Modelling, 18th–20th February

Work Experience

July 1998	KMB GmbH, Spitalgasse 23, 1090 Vienna, Computer Maintenance
Aug. 1999	KMB GmbH, Verterinärplatz 1, 1210 Vienna, Electrical Engineering Technician
Aug. 2000	Schrack BusinessCom, Seybelgasse 13, 1230 Vienna, Computer Network Technician
2001–2008	Schülerhilfe, Wagramer Straße 146, 1220 Vienna, Teacher for Mathematics, Physics and Electrical Engineering
2008–2009	BFI Wien, Alfred–Dallinger–Platz 1, 1034 Vienna, Teacher for Electrical Engineering

Personal Skills and Competences

Language	German (mother tongue), English
Driving license	B
Programming	C, C++, Java
Simulation Environments	MATLAB, Simulink, Maple
Qualifications	Talent of Organisation
	Experience in Team Management and Leadership
	Seminar Certificates in Study Techniques, Rhetoric and Trouble Shooting
Hobbies	Ballroom Dancing, Running, Skiing, Traveling, Theater, Literature