SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

S T U

F C H P T

Efficient Modeling of Hybrid Systems

DIPLOMA THESIS

FCHPT-5414-50907

Bratislava 2012 **Bc. Ján Drgoňa**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

S T U

F C H P T

Efficient Modeling of Hybrid Systems

DIPLOMA THESIS

FCHPT-5414-50907

Study programme:	Automation and Informatization in Chemistry and Food Industry
Study field:	5.2.14 Automation
Workplace:	Linköping University Sweden
Thesis supervisor:	doc. Ing. Michal Kvasnica, PhD.
Consultant:	Dr. Johan Löfberg, PhD.

Bratislava 2012 **Bc. Ján Drgoňa**

Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology
Academic year: 2011/2012
Reg. No.: FCHPT-5414-50907

S T U
F C H P T

DIPLOMA THESIS TOPIC

Student: **Bc. Ján Drgoňa**
Student's ID: 50907
Study programme: Automation and Informatization in Chemistry and Food Industry
Study field: 5.2.14 Automation
Thesis supervisor: doc. Ing. Michal Kvasnica, PhD.

Topic: **Efficient Modeling of Hybrid Systems**

Specification of Assignment:

Objective of the project is to investigate and propose an efficient mathematical framework for modeling of hybrid systems represented as finite-state machines (FSM). Such systems are composed of several local models, interconnected by transitions. It is well known that such interconnections can be modeled by introducing an additional integer variable, whose value indicates which local model is active for the current value of model variables. Traditionally, the integer is encoded using unary encoding, assigning one binary variable to each possible value of the integer. An alternative approach is to model the integer by fewer binary variables using a binary encoding, which only requires log_2(N) binaries. The difficulty of this approach is how to indicate infeasible combinations of bits.

Objectives of the project are as follows:
1) Provide an implementation of a higher-level interface on top of YALMIP for easier modeling of finite state machines.
2) Analyze efficient binary encoding formulations in FSM formulations.
3) Validate obtained results on a case study involving a transportation problem.

Length of thesis: 50
Assignment procedure from: 13. 02. 2012
Date of thesis submission: 19. 05. 2012

L. S.

Bc. Ján Drgoňa
Student

prof. Ing. Miroslav Fikar, DrSc.
Head of office

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Acknowledgement

I would like to express a deep gratitude to my thesis supervisor doc. Ing. Michal Kvasnica, PhD., for his patient supervision, for the given opportunity to study abroad and for his understanding or acceptance of my often excessive jokes and anecdotes.

Also a big gratitude goes to my thesis consultant Dr. Johan Löfberg, PhD., for his substantial contribution to my thesis and for support during my stay in Sweden.

Abstract

This thesis is dealing with modeling of systems containing continuous and discrete dynamic behavior simultaneously. Because of their hybrid nature this kind of systems are called hybrid systems (HS). We highlight several theoretical frameworks for modeling of hybrid systems, at these days most commonly used and well known modeling frameworks are discrete hybrid automata (DHA), piecewise affine (PWA) systems and computational oriented mixed logical dynamical (MLD) systems. Aim of this thesis is to investigate and propose an efficient mathematical framework for modeling of hybrid systems represented either as discrete hybrid automata (DHA) or piecewise affine (PWA) systems. It is well known that such models involving integer or logical variables can be transformed into the corresponding mixed integer programming (MIP) optimization problem. For this purpose we are introducing a technique of Big-M modeling for translating logical statements into equivalent MIP form. Traditionally in corresponding MIP problem the integer is encoded using "unary encoding", assigning one binary variable to each possible value of the integer. Contribution of this thesis lies in alternative approach, where integer variables can be modeled by fewer binary variables using a "binary encoding", which only requires logarithmical number of original binaries $log_2(N)$. The difficulty of this approach is how to indicate infeasible combinations of bits and separate them from feasible region of resulting MIP model by adding extra constraints or so called cuts into the model. Finally in the end of the thesis, we present the implementation of hybrid modeling framework as an extension of free MATLAB optimization toolbox YALMIP, as well as comparison of resulting computational models with models created via modeling language HYSDEL.

Keywords: Hybrid systems, Piecewise affine systems, Finite state machine, Mixed integer programming, Big-M modeling, YALMIP, HYSDEL

Abstrakt

Táto práca sa zaoberá modelovaním takzvaných hybridných systémov (HS), obsahujúcich súčasne spojité aj diskrétne dynamické správanie. Pre modelovanie takýchto systémov bolo vyvinutých viacero prístupov, v prvej časti tejto práce približujeme v súčasnosti najrozšírenejšie a najviac používané teoretické modely, konkrétne ide o diskrétne hybridné automaty (DHA) , po častiach afinné (PWA) systémy a výpočtovo orientované zmiešané logicko-dynamické (MLD) systémy. Cieľom tejto práce je preskúmať a navrhnúť efektívny matematický prístup k modelovaniu hybridných systémov reprezentovaných ako diskrétne hybridné automaty (DHA), alebo po častiach afinné (PWA) systémy. Je všeobecne známe, že takéto modely, obsahujúce celočíselné alebo logické premenné, môžu byť definované vo forme zmiešaného celočíselného (MIP) optimalizačného problému. Pre tento účel predstavujeme techniku takzvaného Big-M modelovania, pomocou ktorej sme schopný transformácie logických výrazov do ekvivalentnej formy problému celočíselného programovania. V probléme zmiešaného celočíselného programovania sú celočíselné premenné klasicky definované pomocou takzvaného unárneho (jednozložkového) kódovania, priraďujúc práve jednu binárnu premennú ku každej možnej celočíselnej hodnote. Príspevok tejto práce spočíva v alternatívnom prístupe kódovania, keď celočíselné premenné môžeme modelovať s použitím menšieho množstva binárnych premenných pomocou takzvaného binárneho kódovania, ktoré vyžaduje len logaritmické množstvo pôvodných binárnych premenných $log_2(N)$. Náročnosť tohto prístupu spočíva v indikácii neriešiteľných kombinácií binárnych premenných a ich separácii z riešiteľnej množiny riešení výsledného modelu pomocou pridania extra ohraničení (tzv. rezov) do príslušného modelu. Na záver predstavujeme programovú realizáciu rozoberaných modelovacích techník ako rozšírenie voľne šíriteľného optimalizačného toolboxu YALMIP pre MATLAB, ako aj porovnanie výsledných modelov s modelmi vytvorenými pomocou modelovacieho jazyka HYSDEL.

Kľúčové slová: Hybridné systémy, Po častiach afinné systémy, Automaty, Zmiešané celočíselné programovanie, Big-M modelovanie, YALMIP, HYSDEL

Contents

List of Symbols and Abbreviations

Symbols, Operators, Functions and Sets

A^T	Transpose of matrix A
k	Time step
N	Set of integers
$\{0,1\}^n$	Set of vectors with *n* binaries
R	Set of real numbers
R^n	Set of real vectors with *n* components
x_r	Real state variable
x_b	Binary state variable
u_r	Real input variable
u_b	Binary input variable
y_r	Real output variable
y_b	Binary output variable
δ_i	Binary auxiliary variable
X_i	Boolean variable
ψ_i	Terms of the product
X_{ij}	Terms of the sum
$\lfloor x \rfloor$	The floor function. Gives the largest integer less than or equal to x
$\lceil x \rceil$	The ceiling function. Gives the smallest integer greater than or equal to x

Abbreviations

CNF	Conjunctive Normal Form
DHA	Discrete Hybrid Automata
EG	Event Generator
FSM	Finite State Machine
LP	Linear Programming

MIP	Mixed Integer Programming
MLD	Mixed Logical Dynamical
MS	Mode Selector
ODE	Ordinary Differential Equation
PWA	Piecewise Affine
SAS	Switched Affine System

Chapter 1

1 Introduction

1.1 Mathematical modeling

Since ancient times people's curiosity led them to exploring the world and try to understand the laws of the nature, but in the past times it was more about philosophy than about real science as we know it today. The first steps for distinguishing the real science from scams and misleading speculations about world was big growth of mathematical knowledge in last few hundred years and since this time mathematical modeling is taking one of the most important role in all scientific fields. The mathematical models are used everywhere, from engineering thru medicine, biology, physics, chemistry, economy and so on. But it is not only matter of science which handles with mathematical modeling, we are surrounded by applications based on mathematical models, which are improving quality of our everyday life.

Mathematical modeling can be conceived as transformation of empirical and practical knowledge of real systems into the theoretical and simplified models of them, by using mathematical language. Unfortunately the real world is still far too complicated for our current mathematical tools to being modeled entirely without any loss of precision, therefore any model is an abstract simplified description of a real system or physical phenomena. Usually there are many ways how to describe a single real-world phenomena, the differences between them are in complexity of particular models and specifications for concrete fields of interest.

Models should be simple enough to formulate an efficient and solvable analysis and synthesis problems for available computational capabilities, but also they should be complicated enough for describing sufficient level of details of the system, what is needed for reliable description of real system. Compromises are needed to be done during the process of the mathematical modeling. The first level of compromise is to identify the most

important parts of the system and include them into the model, the rest less important parts will be excluded due to decreasing the complexity of the model. On the second level of compromise we are taking in mind mathematical methods which are available for solving particular problems. The mathematical procedures for obtaining the model should be elegant and simple enough as the model itself, but also suitable for computer processing and numerical solutions. Actually all previous talk can be expressed in few words by following statements of great man's.

"Make everything as simple as possible, but not simpler."— Albert Einstein

"Challenge in mathematical modeling is not to produce the most comprehensive descriptive model but to produce the simplest possible model that incorporates the major features of the phenomenon of interest." — Howard Emmons

1.2 Classification of dynamical systems

We are using dynamical systems to describe the evolution of some monitored variables, usually states or outputs of the system over time i.e. from their current state to the future state. In the concept of a model of a system are these evolutions traditionally described by differential or difference equations. Therefore most of the theory and tools have been developed for handling such systems as purely continuous or purely discrete, or only in continuous and only in discrete time. Although in recent years a significant need for combining of these two worlds (continuous and discrete) arise from description of some real systems, which are containing both continuous and also logical parts naturally. By combining the continuous and discrete behavior together in single system, the third class of dynamical systems called hybrid systems was born.

Attempt to classify dynamical systems based on the type of their state, appears in the literature [Lys]:

1. **Continuous state**, if the state takes values in Euclidean space R^n for some $n \geq 1$. We will denote $x \in R^n$ as a state of a continuous dynamical system. Demonstration of behavior of continuous state variable is shown on figure 1.1 left.

2. **Discrete state**, if the state takes values in a finite set $\{b_1,\ldots,b_n\}$. We will denote b as a state of a discrete system. Demonstration of behavior of discrete state variable is shown on figure 1.1 right.
3. **Hybrid state variables**, if some of the states takes values in R^n while another states takes values in a finite set. For example, the closed loop system for computer control of an inverted pendulum is hybrid: the state of the pendulum is continuous, while state of the computer is discrete.

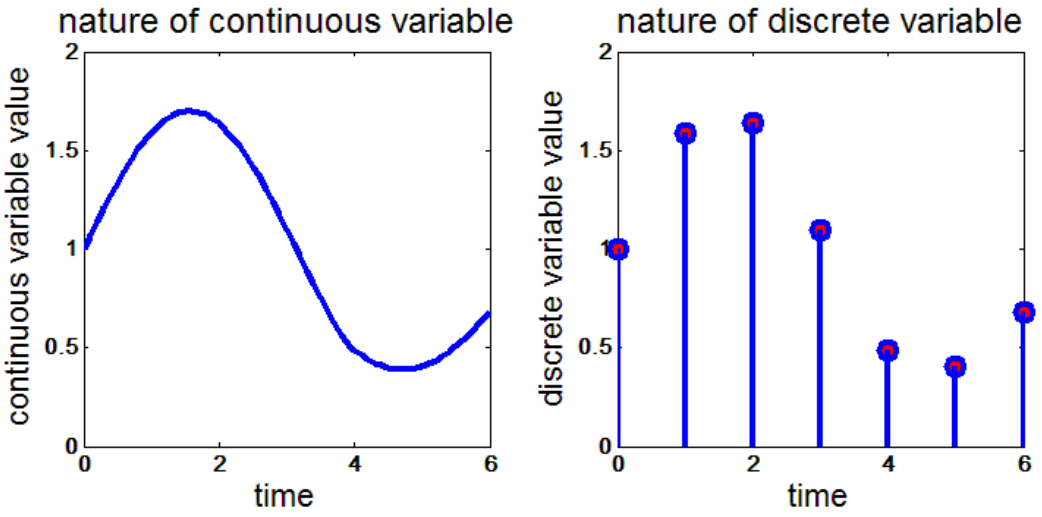

Figure 1.1: Behavior of continuous (left) and discrete (right) variable.

Classification based on the set of times over which the state evolves [Lys]:

1. **Continuous time**, if the set of times can take only real continuous values. We will use $t \in R$ to denote continuous time. The evolution of the state $x(t)$ in a continuous time system is typically described by an ordinary differential equation (ODE). Where $u(t)$ is a vector of inputs in a continuous time and $f\left(x(t),u(t),t\right)$ can be either linear or nonlinear state transition function.

$$\dot{x}(t) = f\left(x(t),u(t),t\right) \tag{1.1a}$$

$$x(t) \in R^n \quad u(t) \in R^m \quad t \in R_{\geq t_0} \quad f : R^{n+m+1} \rightarrow R^n \tag{1.1b}$$

2. **Discrete time**, if the set of times is a subset of the integers. We will use $k \in N$ to denote discrete time. The evolution of the state $x(k)$ in a discrete time system is typically described by a difference equation. Where $u(k)$ is a vector of inputs in a

discrete time and $g\left(x(k),u(k),k\right)$ can be either linear or nonlinear state transition function.

$$x(k+1) = g\left(x(k),u(k),k\right) \tag{1.2a}$$

$$x(k) \in R^n \quad u(k) \in R^m \quad k \in N_{\geq k_0} \quad g : R^{n+m+1} \rightarrow R^n \tag{1.2b}$$

3. **Hybrid time**, when the evolution is over continuous time but there are also discrete moments with special behavior or events.

State systems can be further classified according to the equations used to describe the evolution of their states [Lys]:

1. **Linear**, if the evolution is governed by a linear differential equation (continuous time) or difference equation (discrete time).
2. **Nonlinear**, if the evolution is governed by a nonlinear differential equation (continuous time) or difference equation (discrete time).

1.3 Hybrid Systems

Hybrid models are part of dynamical systems which contains both, continuous and discrete behavior with mutual interactions. Differential or difference equations are used as a typical representation of continuous dynamics, on the other hand discrete part of hybrid systems could be represented by discrete dynamics, logic rules (described by temporal logic, finite state machines, if-then-else conditions, discrete events, etc.) or discrete components (on/off switches, selectors, digital circuitry, software code, etc.). Hybrid systems has many operating modes with different dynamical laws, these modes are switched by mode switches which can be activated by particular state or time events or some external input events [Ant01]. The basic structure of hybrid system is illustrated on Figure 1.2.

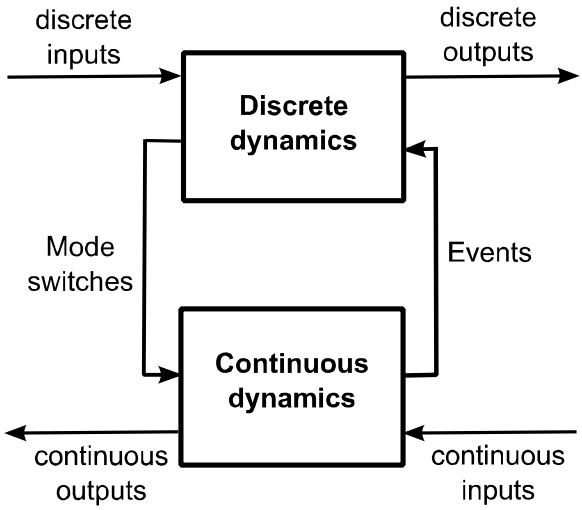

Figure 1.2: Basic structure of hybrid system.

1.3.1 Examples of Hybrid Systems

Hybrid systems are all around us, they arise in a large number of application areas, moreover many physical phenomena admit a natural hybrid description:

- **Mechanical systems:** In these systems the continuous motion may be interrupted by collisions, or they can work in different modes, what can be described as discrete events of finite state machine. The example of such system could be a cruise control system, which controls the transmission gear (discrete input), the engine torque (continuous input), and the braking force (continuous input) in order to track a desired vehicle speed while minimizing fuel consumption and emissions [BemDHS]. Transmissions, stepper motors, and other motion controllers are discussed in literature [Bro01], also as constrained robotic systems [BaGu01]. Another example which shows naturally hybrid behavior is gasoline engine where the power train, gas flow, and thermal dynamics are continuous processes, while the pistons have four discrete operating modes. These systems are logically under deep interest of automotive industry and considerable research in this field was done during recent years more about it can be found in this work [BemDHS] and the references in.
- **Electrical circuits:** Here the continuous phenomena such as charging of capacitors, etc. are interrupted by opening and closing the switches, or diodes going on or off. Into this category belong systems with relays, switches, and hysteresis or computer

disk drives, for more details we recommend to the reader following article [Bran] and its references.

- **Chemical process control:** The continuous evolution of chemical reactions is controlled by discrete actions like opening valves and pumps. More about examples for hybrid modeling of chemical processes could be found in publication [Agar].
- **Embedded computation systems:** When digital computer interacts with a mostly analogue environment. An embedded system is computer system designed for doing specified tasks usually as a part within a bigger system. Examples of these systems we can find everywhere around us, they are taking part in our vehicles, airplanes, factories and so one as shown on a Figure1.3.
- **Networked control systems:** are important class of hybrid systems, where sensing, control, and actuation are not connected directly but they are connected by a shared network medium [ZhBrPh].

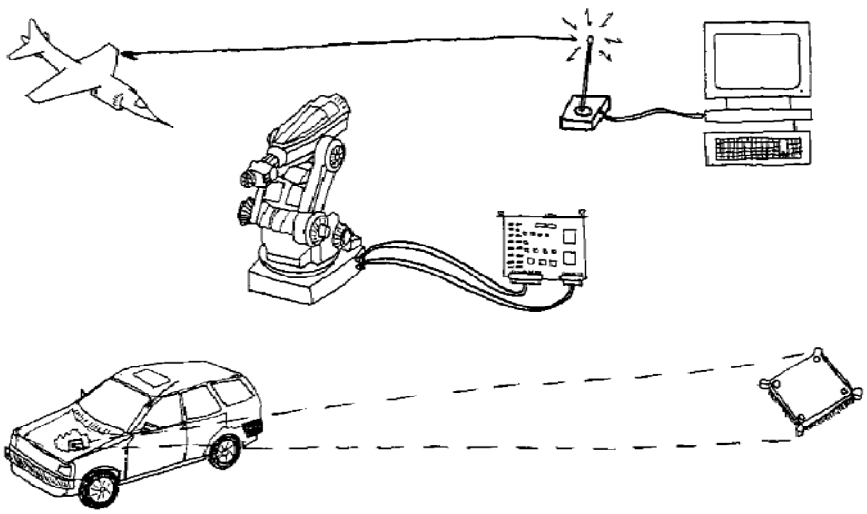

Figure 1.3 [Bran]: Examples of embedded systems.

From theoretical point of view there is a wide range of systems that can be modeled as hybrid systems [Mig, Bran]:

- **Complex systems:** organized in hierarchical way, where for example discrete planning algorithms at the higher level interact with continuous control algorithms and processes at the lower level [BemDHS]. In engineering practice there was few

attempts to model complex systems like automated highway systems (AHSs) [Lys] or multi-vehicle formations and coordination [Olaru].

- **Multiple model systems:** These are systems which general model and their overall evolution is governed by different sub-models, by partitioning the state space into regions with assigned sub-models (e.g. piecewise affine systems) or by changing system parameters according to a given signal (e.g. switched systems or systems with operating mode changes). Applications of these models appear in engineering practice for example in flight control and air traffic management systems [Bran, LysTom].
- **Systems with switching components:** Systems in this category include switching elements like relays, dead-zones or hysteresis, more about these examples can be found again in paper [Bran] and its references. Therefore electrical circuits could be considered as one of the real world example for these types of hybrid systems.
- **Adaptive systems:** The hybrid nature of these systems lies in switching rules, provided e.g. by piecewise affine systems or by finite state machines governing the adaptation law.
- **Systems with modeled failures:** In case of sudden or abrupt faults, the occurrence of a failure in a system can be modeled as a switching signal. The fault-prone system can be then considered as a hybrid system.
- **Systems involving synchronization signals:** Such systems arise e.g. in communication networks.

Even if we formally divided hybrid systems in some specific subclasses and types, it is important to note, that all kind of hybrid systems are deeply interconnected and equivalent in their nature. A single real process could be classified as a member of different hybrid model classifications in the same time.

1.3.2 Motivation for Hybrid Systems

Motivation for initiation of the research and introduction of theoretical fundamentals for hybrid systems lied in the fact, that there was no known single model capable of capturing discrete and continuous dynamics together. Efficient tools for modeling analysis and synthesis of hybrid systems was developed only recently. The theory of hybrid systems is

connecting contributions from continuous system and control theory with field of computer science called discrete event system theory. Connecting these two on first look different and non-connected fields of engineering science was a big challenge. It was crucial to combine capabilities of different modeling frameworks to be able to describe the behavior of hybrid systems.

The design and analysis of hybrid systems are in general more difficult than design and analysis of only discrete or only continuous systems, this is because the discrete dynamics is affecting the continuous evolution and vice versa [Lys]. The interconnections between discrete and continuous behavior are mostly very tight, therefore in modeling of discrete-continuous relations is common to represent the discrete events as instant changes in continuous dynamics.

Because of this in most practical cases, the synthesis of control schemes for systems having also a discrete and continuous dynamical nature is still approached with heuristic rules, usually driven by engineering insight and experience, but consequently this approach requires long design and verification time. The interest of the control community is motivated by several clearly apparent trends in industry which is calling for creating new tools to design control schemes for hybrid systems and to analyze their stability, safety, and performance. Based on these needs several problems are currently investigated in the theory of hybrid systems, it is the definition and computation of trajectories, stability and safety analysis, control, state estimation, etc. [BemDHS].

A simulator is usually used for definition of trajectories of hybrid systems, in general it is a mathematical prediction model made to compute the time evolution of the variables of the system, through the simulations we are able to verify and probe the correctness of the model of the system. Tools like reachability analysis or piecewise quadratic Lyapunov stability becomes a standard procedures in analysis of hybrid systems, more about this can be found in literature e.g. [BemDHS, Hys, Mig, Tor] and their references. For controlling of a hybrid model most of the nonlinear and logically also linear control theory can not be applied because of special behavior of hybrid systems. At these days as most commonly used approach for control of hybrid systems is an optimal control theory, which foundations was laid by Richard Bellman and Lev Pontryagin.

1.3.3 Modeling frameworks

Several modeling frameworks have been proposed recently to represent hybrid systems. Each modeling class is usually made for dealing with particular problems, and therefore they seem to be dissimilar at first look. But recent research shows that all hybrid modeling classes are equivalent and therefore the models created in different framework can be under additional assumptions transformed into another model framework. Therefore the same system can be represented with models of each class. This is very important acquaintance which allows us to choose most convenient hybrid modeling framework for concrete problems. The equivalence of hybrid modeling classes has been proved for example in paper [Equival] or can be found also in works [Mig, HeSB01].

One of the earliest attempts for creating hybrid modeling framework appeared in the process literature is called the theory of differential algebraic equations (DAEs) with an index set, used as possible discrete model of a system [BarPan01].

In literature most commonly used frameworks are timed automata [Silv01, Asar01] and hybrid automata [Silv01, Al01, AlDi01]. Automata are elegant frameworks for modeling hybrid systems and become very popular and proved to be successful for formal verification of the models. We will introduce the discrete hybrid automata (DHA) [BemDHA, Hys, Mig, Tor] later in Chapter 2.

At these days as most important and well known modeling subclasses we can further mention are e.g. Mixed logical dynamical (MLD) models [BemMor, BempDHA, Tor, Mig], piecewise affine (PWA) systems [Sontag], linear complementarity (LC) systems [HeSchW, CaHeS, He01, AJSMS01, AJSMS02], extended linear complementarity (ELC) systems [SchM02, Equival], max-min-plus-scaling (MMPS) systems [SchB01, Equival], first-order linear hybrid systems with saturation [Sch01] and linear coupled component automata (LCCA) [Agar].

Each modeling subclass has its own specifications and advantages compared the others. For example, control and verification techniques as reachability/observability analysis for MLD hybrid models, stability criteria were proposed for PWA systems [BeToMo], and conditions of existence and uniqueness of solution trajectories (well-posedness) for LC systems and so on.

1.3.4 Future of Hybrid Systems

As is apparent from previous talk hybrid models are highly demanding on computation power due to their complexity. It is important to point that modeling of hybrid systems is creating mathematical problems which belongs to so called group of NP-hard problems [wikiNPh], what means that computational time may grow exponentially with dimension (number of variables) of the problem in worst case. Computational tools sufficient for dealing with this type of problems become available only recently, what has brought a huge space of opportunities for exploring and applications of hybrid systems. Therefore hybrid systems become currently very popular and important field of study among both academic and industrial researchers.

Ideal theoretical visions to the future are talking about systems as whole entities without heterogeneous parts [AnNe]. So there will be no need for denoting system to be discrete, continuous or hybrid, it will be just a system describing and incorporating entire dynamics by uniform rules.

1.4 Basic Terminology and Definitions

1.4.1 Convex Set

Definition 1.1 [BoyVan]: A set $C \subset R^n$ is *convex* if the line segment between any two points in C lies in C, i.e. for any $x_1, x_2 \in C$ and any real number θ, where $0 \leq \theta \leq 1$, is true

$$\theta x_1 + (1-\theta) x_2 \in C \tag{1.1}$$

Roughly speaking a set C is convex if any two points lying in the set $x_1, x_2 \in C$, can be connected by a straight line which lies entirely within the set C. Examples of convex and nonconvex sets are shown on following figure 1.4.

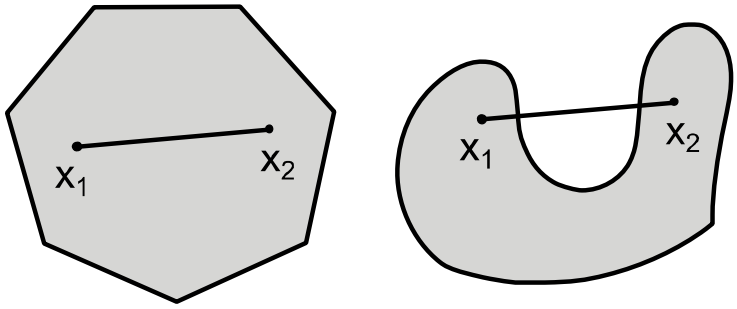

Figure 1.4: *Left* - example of convex set. *Right* - example of nonconvex set.

1.4.2 Convex Hull

Definition 1.2 [BoyVan]: The convex hull of a set C, denoted $conv\,C$, is the set of all convex combinations of points $x_1,\ldots,x_k \in R^n$ in C:

$$conv\,C = \{\theta_1 x_1 + \ldots + \theta_k x_k \mid x_i \in C,\ \theta_i \geq 0,\ \theta_1 + \ldots \theta_k = 1,\ i = 1,\ldots,k\} \tag{1.2}$$

The convex hull is always convex, it is the smallest convex set of any set B, therefore $conv\,B \subseteq B$. Examples of convex hulls are shown on figure 1.5.

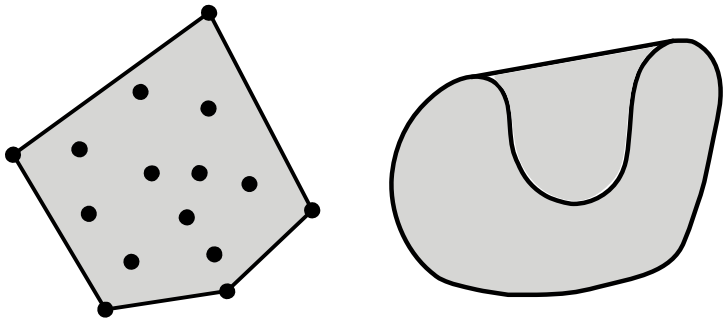

Figure 1.5: *Left* - Convex hull of set of 15 points. *Right* – convex hull of nonconvex set.

1.4.3 Polytope

In literature a polytope has two standard representations, the *V-representation* and *H-representation*. All polytopes are convex sets.

Definition 1.3 [BoyVan]: The *H-representation* of polytope is defined as a solution set (1.3a) of linear inequalities and equalities (1.3b). Where $x \in R^n$, and matrixes A, B, C, D are containing real coefficients for linear inequalities and equalities.

$$P = \{x \mid Ax \leq B,\ Cx = D\} \tag{1.3a}$$

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_m^T \end{bmatrix},\ B = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix},\ C = \begin{bmatrix} c_1^T \\ \vdots \\ c_p^T \end{bmatrix} \text{ and } D = \begin{bmatrix} d_1 \\ \vdots \\ d_p \end{bmatrix} \tag{1.3b}$$

Definition 1.4 [BoyVan]: The *V-representation* of polytope is defined as convex hull (1.4) of finite number of points in Euclidean n-dimensional space: $x_1, \ldots, x_k \in R^n$.

$$P = conv(x_1, \ldots, x_k) \tag{1.4}$$

A vertex of polytope, which all components are integers, is called an integral vertex.

1.4.4 General Optimization Problem

Mathematical optimization problem has a form

$$\text{minimize:} \quad f(x) \tag{1.5a}$$

$$\text{subject to:} \quad g_i(x) \leq b_i, \quad i = 1, \ldots, m \tag{1.5b}$$

$$h_j(x) = k_j, \quad j = 1, \ldots, p \tag{1.5c}$$

Where the vector $x = (x_1, \ldots, x_n)$ is the optimization variable of the problem, $f(x): R^n \to R$ is the objective function, $g_i(x): R^n \to R$ are representing inequality constraint functions, $h_j(x): R^n \to R$ are representing equality constraint functions, and the constants $b_i, k_j \in R$ are bounds for constraints. The vector $x*$ is called an optimal solution of the problem (1.5), if it has smallest value of objective function (1.5a) while holding the constraints (1.5b,c).

1.4.5 Linear Programming

Linear programming (LP) is an important class of optimization problems, in which objective function and all constraints are linear.

$$\text{minimize:} \quad c^T x + d \tag{1.6a}$$

$$\text{subject to:} \quad a_i^T x \le b_i, \quad i = 1,\dots,m \tag{1.6b}$$

$$h_j^T x = k_j, \quad j = 1,\dots,p \tag{1.6c}$$

Where the vectors $c, a_i, h_j \in R^n$ and scalars $d, b_i, k_j \in R$.

1.4.6 Mixed Integer Programming

Mixed-integer programming (MIP) is another important class of optimization problems, which characteristic hallmark is that they are containing both, real and also integer valued variables. We are talking about mixed-integer linear programming (MILP) problem, when the objective function and constraints are linear.

$$\text{minimize:} \quad f(x) \tag{1.7a}$$

$$\text{subject to:} \quad g_i(x) \le b_i, \quad i = 1,\dots,m \tag{1.7b}$$

$$h_j(x) = k_j, \quad j = 1,\dots,p \tag{1.7c}$$

$$x_l \in N \quad l = 1,\dots,p \tag{1.7d}$$

Where x_l denotes integer variables.

Because of this mixed nature, MIP problems are suitable for capturing hybrid dynamics and therefore for modeling of hybrid systems. In this thesis we are dealing with modeling frameworks which are constructing representing MIP problems for description of hybrid system.

Chapter 2

2 Modeling of Hybrid Systems

Efficiency is the most important feature for solving of optimization problems, but for obtaining efficient solution not only high performance solvers are required, also modeling issues are playing very important role in optimization process. Different modeling approaches are providing models with various complexities, and even small changes in the models structure can cause huge improvement of optimization efficiency. Especially in case of hybrid models which could be extremely complex and hard to solve, efficiency is crucial task for each modeling framework.

In this chapter we will describe several hybrid model representations introduced in literature [BemDHS, BemMor, Tor, Mig]. Particularly we are dealing with discrete hybrid automatas (DHA), piecewise affine systems (PWA) and mixed logical dynamical systems (MLD).

2.1 Model Classes for Hybrid Systems

As we mentioned in Chapter 1 there is a big amount of divisions and classifications of hybrid systems. For our purposed is not necessary to mention or describe all of them, in this section we will focus only on three well known modeling classes of hybrid systems, with which we are dealing in this work and therefore it will be convenient to spare some words about them. In following pages we will show how easy it is to describe hybrid systems as discrete hybrid automatas (DHA). These models can be conceived as general modeling representation for hybrid systems, and consequently can be easily transformed into other classes of hybrid systems. DHAs are powerful tool for description of hybrid systems, but due to their hybrid and autonomous nature they are not suitable for control and properties investigation of modeled systems. Next modeling framework what we will discuss are piecewise affine systems (PWA). These models can be widely used for example as approximation of nonlinear functions, and can be easily transformed into corresponding mixed-integer optimization problems. The last modeling framework is called mixed logical dynamical systems (MLD),

models created within this framework are computation oriented, because they are internally defined in form of mixed-integer inequalities, and therefore are suitable for solving analysis, optimal control, and receding horizon estimation problems.

2.2 Discrete Hybrid Automata

A discrete hybrid automaton (DHA) is a dynamical system that describes the evolution in time of the values of a set of discrete and continuous state variables [Lys]. The model is called hybrid because it combines nonlinear continuous dynamics with the dynamics of discrete event systems. Continuous part of DHA is represented by switched affine systems (SAS) which are described by a set of ordinary differential equations and discrete dynamics of the systems is represented as finite state machine (FSM). Additional elements of DHA are the event generator (EG) and the mode selector (MS) which provides the interactions between discrete (FSM) and continuous part of the system (SAS). The EG extracts and generates logic signals from the continuous part of the system, this is done in form of non/satisfying of the linear-thresholds for continuous variables (states, inputs, outputs). Those logic events and other exogenous logic inputs trigger the switch of the state of the FSM. Then the MS is processing all logic signals (states, inputs, time events, linear-thresholds) to choose corresponding mode of continuous dynamics for SAS. Block diagram representation of DHA is shown on figure 2.1.

We are dealing with DHA models because they are fairly rich in descriptive power, also compilation of such models for description of real systems are usually very intuitive and easy to do, because of these properties DHA is very popular and widely used as a modeling framework for hybrid systems among academic and engineering society.

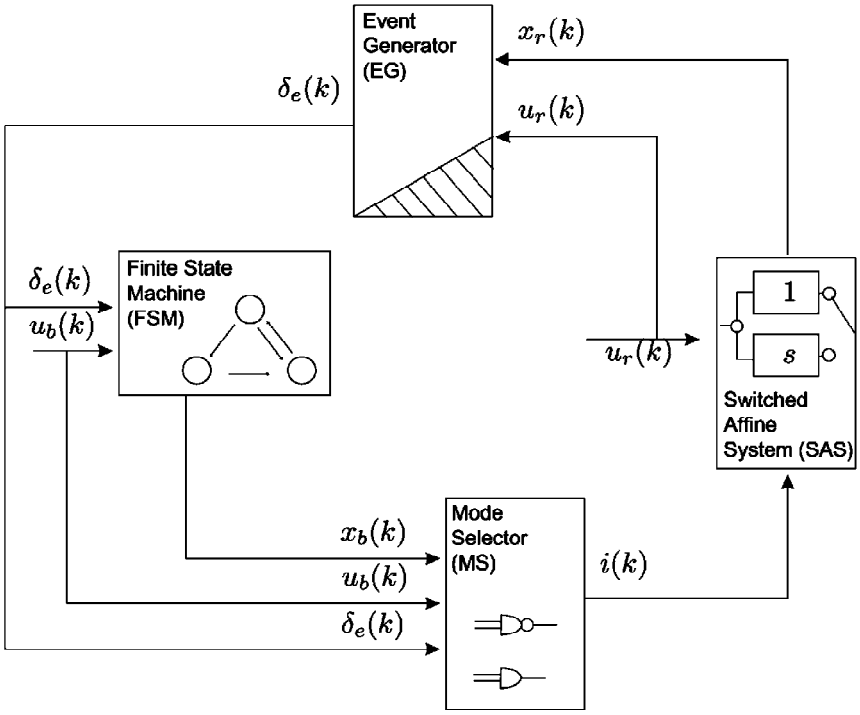

Figure 2.1 [ToBe]: Representation of DHA as a connection of EG, FSM, SAS and MS.

2.2.1 Switched Affine Systems

A switched affine system (SAS) is a collection of linear affine systems:

$$x_r(k+1) = A_i x_r(k) + B_i u_r(k) + f_i(k) \tag{2.1a}$$

$$y_r(k) = C_i x_r(k) + D_i u_r(k) + g_i(k) \tag{2.1b}$$

Where $k \in N^+$, $x_r \in X_r \subseteq R^n$ is the continuous state vector, $u_r \in U_r \subseteq R^m$ is the exogenous continuous input vector, $y_r \in Y_r \subseteq R^p$ is the continuous output vector, $\{A_i, B_i, f_i, C_i, D_i, g_i\}$ are the matrices of suitable dimensions, and the mode $i(k) \in I := \{1,..,s\}$ is an input signal that chooses the linear state update dynamics. When a switch occurs a SAS of the form (2.1) preserves the value of the state, however it is possible to implement reset maps on a SAS. The reset can be taken as a special dynamics that is active only for one sampling step. With using reset maps we are able to model also non continuous dynamics or functions. A SAS can be rewritten as the combination of linear terms and *if-then-else* rules: the state-update Equation (2.1a) is equivalent to

$$z_1(k) = \begin{cases} A_1 x_r(k) + B_1 u_r(k) + f_1(k), & if\ (i(k) = 1) \\ 0, & otherwise \end{cases} \tag{2.2a}$$

$$\vdots$$

$$z_s(k) = \begin{cases} A_s x_r(k) + B_s u_r(k) + f_s(k), & if\ (i(k) = s) \\ 0, & otherwise \end{cases} \tag{2.2b}$$

$$x(k+1) = \sum_{i=1}^{s} z_i(k) \tag{2.2c}$$

where $z_i(k) \in R^{n_r}$, $i = 1, \ldots, s$, and (2.1b) admits a similar transformation.

2.2.2 Event Generator

An event generator (EG) is a mathematical object that generates a logic signal according to the satisfaction of a linear constraint:

$$\delta(k) = f_H\left(x_r(k), u_r(k), k\right) \tag{2.3}$$

Where $f_H : R^n \times R^m \times Z^+ \to D \subseteq \{0,1\}^n$ is a vector of descriptive functions of a linear hyperplane, and $Z^+ := \{0,1,\ldots\}$ is the set of nonnegative integers. In particular, time events are modeled as: $\left[\delta^i(k) = 1\right] \leftrightarrow \left[kT_s \geq t_0\right]$, where Ts is the sampling time, while threshold events are modeled as: $\left[\delta^i(k) = 1\right] \leftrightarrow \left[a^T x_r(k) + b^T u_r(k) \leq c\right]$, where the superscript *i* denotes the *i*-th component of a vector.

2.2.3 Finite State Machine

A finite state machine (FSM) or automaton is a discrete dynamic process that evolves according to a logic state update function:

$$x_b(k+1) = f_B\left(x_b(k), u_b(k), \delta(k)\right) \tag{2.4a}$$

where $x_b \in X_b \subseteq \{0,1\}^{n_b}$ is the Boolean state, $u_b \in U_b \subseteq \{0,1\}^{u_b}$ is the exogenous Boolean input $\delta(k)$ is the endogenous input coming from the EG, and

$f_B : X_r \times U_r \times D \to X_b$ is a deterministic logic function. A FSM can be conveniently represented using an oriented graph. A FSM may also have an associated Boolean output

$$y_b(k) = g_B\left(x_b(k), u_b(k), \delta(k)\right) \tag{2.4b}$$

where $y_b \to Y_b \subseteq \{0,1\}^{p_b}$ and $g_B : X_r \times U_r \times D \to Y_b$. The idea of transforming a well-posed FSM into a set of Boolean equalities was already presented in [Parbar01] where the authors performed model checking using (mixed) integer optimization on an equivalent set of integer inequalities.

2.2.4 Mode Selector

The mode selector (MS) consist of the logic state $x_b(k)$, the Boolean inputs $u_b(k)$, and the events $\delta(k)$ which select the dynamic mode $i(k)$ of the SAS through a Boolean function $f_M : X_b \times U_b \times D \to I$. The output of this function $i(k)$,

$$i(k) = f_M\left(x_b(k), u_b(k), \delta(k)\right) \tag{2.5}$$

is called active mode. Literature says that a mode switch occurs at step k if $i(k) \neq i(k-1)$.

In this discrete-time setting a mode switch can only occur at sampling instants, contrarily to continuous time hybrid models, where switches can occur at any time.

2.2.5 DHA Trajectories

For the given initial condition $\left[x_r(0); x_b(0)\right] \in X_r \times X_b$, and for the input $u(k) = \left[u_r(k); u_b(k)\right] \in U_r \times U_b$, $k \in N^+$ the state trajectory $x(k), k \in N^+$, of the system is recursively computed as follows:

1. Initialization: $x(0) = \left[x_r(0); x_b(0)\right]$;
2. Recursion:
   a. $\delta(k) = f_H\left(x_r(k), u_r(k), k\right)$
   b. $i(k) = f_M\left(x_b(k), u_b(k), \delta(k)\right)$
   c. $y_r(k) = C_i x_r(k) + D_i u_r(k) + g_i(k)$
   d. $y_b(k) = g_B\left(x_b(k), u_b(k), \delta(k)\right)$
   e. $x_r(k+1) = A_i x_r(k) + B_i u_r(k) + f_i(k)$
   f. $x_b(k+1) = f_B\left(x_b(k), u_b(k), \delta(k)\right)$

Definition 2.1 [BemDHS, Tor]: A DHA is well-posed on $X_r \times X_b$, $U_r \times U_b$, $Y_r \times Y_b$, if for all initial conditions $x(0) = [x_r(0); x_b(0)] \in X_r \times X_b$, and inputs $u(k) = [u_r(k); u_b(k)] \in U_r \times U_b$, $k \in N^+$, the state trajectory $x(k) \in X_r \times X_b$, and output trajectory $y(k) = [y_r(k); y_b(k)] \in Y_r \times Y_b$, are uniquely defined.

This definition can be used also for other types of hybrid models what were introduced before. In general a hybrid model may not be well-posed, either because the trajectories stops after a finite time or because of nondeterminism (the successor $x(k+1)$ may be multiply defined) [BemDHS,Tor]. But note that trajectories of DHA are deterministic, therefore also well-posed.

2.3 Piecewise Affine Systems

A particular case of DHA is the popular class of piecewise affine (PWA) systems [HeeSchBem, Son, FTMLM03, RBL04]. PWA systems are for short defined by partitioning the space of states and inputs into polyhedral regions and associating with each region a different linear state-update equation. Essentially, PWA are switched affine systems whose mode only depends on the current location of the state vector.

$$x(k+1) = A_i x(k) + B_i u(k) + f_{i(k)} \tag{2.6a}$$

$$y(k) = C_i x(k) + D_i u(k) + g_{i(k)} \tag{2.6b}$$

For $[x(k); u(k)] \in \chi_i$, where $x_r \in X_r \subseteq R^n$, is the state, $u \in U \subseteq R^m$, is the input and $y \in Y \subseteq R^p$ is the output at time instance k. $\{\chi_i\}_{i=1,\ldots,s}$ is a polyhedral partition of the state-input space defined by a system of inequalities $\{H_x^i x + H_u^i u \le K^i\}$ and $\{A_i, B_i, f_i, C_i, D_i, g_i, H_x^i, H_u^i, K^i\}$, are real matrices of suitable dimensions.

PWA systems are the “simplest” extension of linear systems that can still model non-linear and non-smooth processes with arbitrary accuracy and are capable of handling hybrid phenomena. For PWA systems, *well-posedness* [wikiWP01, wikiWP02] is defined as follows.

Definition 2.1 [BemDHS]: A PWA system is *well-posed* on (X,U,Y), if for all initial conditions $x(0)=X$ and for all inputs $x(k)\in X$, for all $k\in N$, the state trajectory $x(k)\in X$ and the output trajectory $y(k)\in Y$ are uniquely defined.

When the mode $i(k)=\{1,\ldots,s\}$ is an exogenous variable, the condition $[x(k);u(k)]\in\chi_i$ disappears and we refer to (2.6) as a switched affine system (SAS).

On figure 2.2 there is shown example of PWA system defined by one variable PWA function divided into five regions, which are including different dynamical laws.

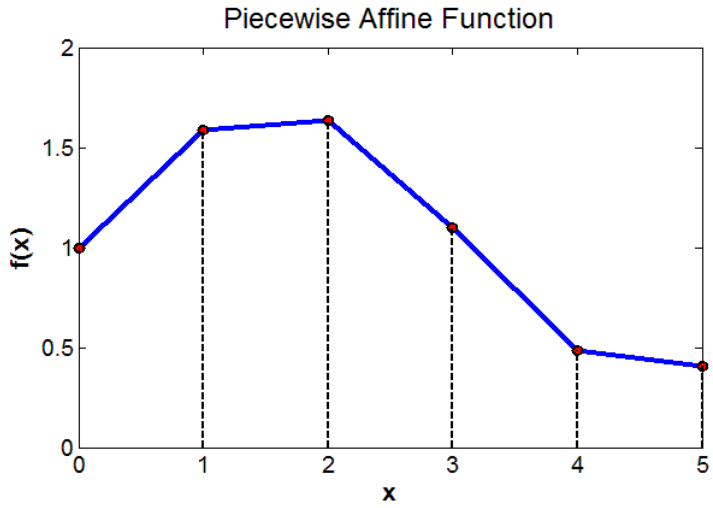

Figure 2.2: Example of one dimensional PWA function with 5 regions.

2.4 Mixed Logical Dynamical Systems

In [BemDHS, BemMor] a class of hybrid systems, called Mixed Logical Dynamical (MLD) systems, has been introduced in which logic, dynamics and constraints are integrated. An MLD system is described by the following relations:

$$x(k+1)=Ax(k)+B_1u(k)+B_2\delta(k)+B_3z(k) \tag{2.7a}$$

$$y(k)=Cx(k)+D_1u(k)+D_2\delta(k)+D_3z(k) \tag{2.7b}$$

$$E_2\delta(k)+E_3z(k)\le E_1u(k)+E_4x(k)+E_5 \tag{2.7c}$$

Where $x(k) = \left[x_r^T(k) \;\; x_{br}^T(k)\right]^T$ with states of the system $x_r(k) \in R^{n_r}$, $x_b(k) \in \{0,1\}^{n_b}$, outputs $y(k)$ and inputs $u(k)$ have similar structure as states. Next $z(k) \in R^{r_r}$ are real and $\delta(k) \in \{0,1\}^{r_b}$ binary auxiliary variables and $A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1 \dots E_5$, are real matrices of suitable dimensions. Auxiliary variables are introduced when translating propositional logic or PWA functions into linear inequalities. All constraints for variables of new MLD system are summarized in the mixed-integer linear inequality constraint (2.7c). The MLD system is considered to be *completely well posed* if for a given state $x(k)$ and input $u(k)$ the values of $\delta(k)$ and $z(k)$ uniquely defined by the inequality (2.7c). A formal definition of wellposedness for MLD systems and a algorithm to test the well-posedness have been presented in [BemMor].

The MLD framework is a powerful tool for modeling discrete-time linear hybrid systems, it aims at translating a hybrid system in a set of mixed integer linear equalities and inequalities. Via MLD framework we are able to describe automata, propositional logic, if-then-else statements and PWA functions. General nonlinear functions cannot be modeled and have to be approximated by PWA functions.

Chapter 3

3 Hybrid Systems Modeling Framework

3.1 Introduction

Aim of this Chapter is on investigating and proposing an efficient mathematical framework for modeling of hybrid systems. The core of this framework will be build on translation techniques for efficient rearrangement of systems with hybrid dynamics or logical components defined either as DHA or PWA into corresponding mixed-integer problem, what is suitable computational representation of hybrid systems solvable by numerical solvers.

At the beginning of this Chapter we are investigating the connections between logic propositions and mixed-integer linear constraints. First we will highlight a general propositional calculus for handling the Boolean variables, and then we will focus on techniques for translation arbitrary Boolean statements or functions into equivalent mixed-integer linear inequalities. We will also introduce the Big-M conversion, which is efficient tool for converting a complex, possibly nonconvex or logical constraints and functions into form of mixed-integer inequalities. Later on we will show how to transform more complex hybrid systems defined as DHA or PWA into MIP form by using Big-M formulations with combination of translation techniques defined for Boolean functions. And in the end of this Chapter we are proposing an attempt to make these general transformation techniques more efficient by decreasing logarithmically the number of auxiliary binary variables included in resulting mixed-integer problem. This can be done by “binary encoding” of integer variables in original model by new auxiliary binary variables. For the purpose of improvement of efficiency and solvability of resulting model we are also presenting the cutting-plane approach to eliminate infeasible combinations of new auxiliary binary variables with focus on minimization of number necessary cuts i.e. extra linear constraints included in our model.

3.2 Logical Propositions

In the beginning we will start with some basic definitions of Boolean algebra. A variable X is referred to as a Boolean variable or literal, if $X \in \{0, 1\}$. Boolean algebra enables statements to be combined in compound statements by logical operators named in following Table 3.1.

Logical operator	**Symbol**
Logical conjunction - AND	$\wedge$
Logical disjunction – OR	$\vee$
Logical negation - NOT	$\neg$
Logical implication - IF	$\rightarrow$
Logical equivalence – IF AND ONLY IF	$\leftrightarrow$
Logical exclusive or - XOR	$\oplus$

Table 3.1: Logical operators and their characters.

Logical operators or connectives have several properties [Chr01] whose allows to transform compound statements into equivalent statements by using different connectives and simplify complex statements. It is known that all connectives can be defined in terms of a subset of them, which is said to be a complete set of connectives $\{\vee, \neg\}$ [BemMor]. In literature a minimal representation of Boolean statement is called conjunctive normal form (CNF), where the only propositional connectives a formula in CNF can contain are AND, OR, and NOT $\{\wedge, \vee, \neg\}$ [wikiCNF]. The Boolean statement *F* is in CNF (*Boolean equivalent of products of sums*) if it is written in following form:

$$F = \bigwedge_{i=1}^{n} \psi_i \tag{3.1a}$$

$$\psi_i = \bigvee_{j=1}^{m} X_{ij} \tag{3.1b}$$

Where the Boolean formulas ψ_i are named *terms of the product*, and X_{ij} are named *terms of the sum*. The formula is in minimal CNF when the formula has minimum number of terms

of the product and each term has the minimum number of terms of sum [BemDHS]. Every Boolean expression can be rewritten as a minimal CNF [Koh]. Logical expressions which are not part of CNF $\{\wedge, \vee, \neg\}$ or complete set of connectives $\{\vee, \neg\}$, can be rewritten into these forms by using logical equivalences between logical expressions as is shown in following example 3.1. More about logical equivalences you can find in papers [BemMor, Mig] and their references.

Example 3.1:

Logical statement	Equivalent logical statement
$X_1 \wedge X_2$	$\neg(\neg X_1 \wedge \neg X_2)$
$X_1 \to X_2$	$\neg X_2 \to \neg X_1$
$X_1 \to X_2$	$\neg X_1 \vee X_2$
$X_1 \leftrightarrow X_2$	$(X_1 \to X_2) \wedge (X_2 \to X_1)$

Table 3.2: Example for equivalency of logical statements.

When a Boolean expression is used to define a Boolean variable X_n as a function of $X_1, \ldots, X_{n-1}$, it is also called a Boolean function f defined as follows.

$$X_n = f(X_1, X_2, \ldots, X_{n-1}) \tag{3.2a}$$

Relations between Boolean variables $X_1, \ldots, X_n$, can be defined with Boolean formula F.

$$F(X_1, X_2, \ldots, X_n) = 1 \tag{3.2b}$$

Where $X_i \in \{0, 1\}, i = 1, \ldots, n$. Each Boolean function is also a Boolean formula, but this statement is not valid conversely i.e. each Boolean formula doesn't have to be a Boolean function. Boolean formulas can be equivalently translated into a set of mixed-integer inequalities (MIP), what we will show in next stage of this thesis.

Complex theory of Boolean calculus can be found in digital circuit design texts, e.g. [Chr01, Hay01]. And more mathematically rigorous interpretation can be found e.g. [Med01].

3.3 Propositional Calculus and Mixed-Integer Programming

In this section we are presenting several general techniques for translation of logical statements into computable form of mixed-integer inequalities. Incentive to do so is that mixed-integer programming problem has been advocated as an efficient tool to perform automated deduction of validity of logical propositions [CaPaSo]. For further reading about techniques of translation process and generalization of some results we recommend to the reader following works [BempDHS, Mig] and their references.

First we want to point on conversion of basic logical statements into mixed-integer inequalities what is shown in Table 3.2. Let's associate with Boolean variable X_i a logical variable $\delta_i \in \{0,1\}$ which has a value of 1 if $X_i = T$, and 0 if $X_i = F$.

Operator	Logical statement	Mixed-integer (in)equality
AND	$X_1 \wedge X_2$	$\delta_1 = 1, \delta_2 = 1, \quad or \quad \delta_1 + \delta_2 \geq 2$
OR	$X_1 \vee X_2$	$\delta_1 + \delta_2 \geq 1$
NOT	$\neg X_1$	$\delta_1 = 0, \quad or \quad 1 - \delta_1 \geq 1$
XOR	$X_1 \oplus X_2$	$\delta_1 + \delta_2 = 1$
IMPLIES	$X_1 \rightarrow X_2$	$\delta_1 - \delta_2 \leq 0$
IFF	$X_1 \leftrightarrow X_2$	$\delta_1 - \delta_2 = 0$

Table 3.3: Conversion of basic logical statements into mixed-integer inequalities.

In literature [BemDHS, Mig] there are mentioned two general methods for conversion of logical statements into mixed-integer inequalities. Authors are naming them symbolical and geometrical method, further we are extending symbolical method to be more general and applicable on any type of logical statements, and using it as a main translation technique for our modeling framework. But it is important to mention that all these techniques are in the end proposing equivalent results, because they are tracking the same objective, which is to find equivalent mixed-integer linear inequalities for arbitrary Boolean functions (3.2a) or

formulas (3.2b). Therefore no method is uniformly better than the others and the choice of a suitable method is dependent on the form of the logical statements.

3.3.1 Symbolical Method

Symbolical or CNF method is based at first on transforming Boolean functions (3.2a) or Boolean formulas (3.2b) into conjunctive normal form (CNF). This can be done automatically by using one of the standard well known techniques mentioned in [ChHoo01, Chr01]. Let us consider to have the CNF defined as following.

$$\bigwedge_{j=1}^{m}\left(\bigvee_{i\in P_j} X_i \bigvee_{i\in N_j} \neg X_i\right) \tag{3.3a}$$

Where $N_j, P_j \subseteq \{1,\ldots,n\}, \forall j = 1,\ldots,m$.

This CNF than can be transformed into the mixed-integer inequalities with corresponding binary variables δ_i like this

$$\begin{aligned} 1 &\le \sum_{i\in P_1}\delta_i + \sum_{i\in N_1}(1-\delta_i), \\ &\vdots \\ 1 &\le \sum_{i\in P_m}\delta_i + \sum_{i\in N_m}(1-\delta_i). \end{aligned} \tag{3.3b}$$

3.3.2 Extended Symbolical Method

Disadvantage of previous symbolical method is demand for logical statements to be in CNF. Therefore we will present extended symbolical method used in this thesis, which is able to use full scale of logical operators, not only CNF. In the Table 3.4 there are shown few most commonly used examples of general logical statements transformed into MIP inequalities via extended symbolical method.

Logical statement	Mixed-integer (in)equality
$X_0 \leftarrow \neg X_1$	$\delta_0 \ge 1-\delta_1$
$X_0 \rightarrow \neg X_1$	$\delta_0 \le 1-\delta_1$
$X_0 = \neg X_1$	$\delta_0 = 1-\delta_1$

$X_0 \leftarrow X_1 \wedge X_1 \wedge \ldots X_n$	$\delta_0 \geq 1 + \sum_{i=1}^{n} \delta_i - n$
$X_0 \rightarrow X_1 \wedge X_1 \wedge \ldots X_n$	$\delta_0 \leq \delta_i$
$X_0 = X_1 \wedge X_1 \wedge \ldots X_n$	$\delta_0 \geq 1 + \sum_{i=1}^{n} \delta_i - n \ , \ \delta_0 \leq \delta_i$
$X_0 \leftarrow X_1 \vee X_1 \vee \ldots X_n$	$\delta_0 \geq \delta_i$
$X_0 \rightarrow X_1 \vee X_1 \vee \ldots X_n$	$\delta_0 \leq \sum_{i=1}^{n} \delta_i$
$X_0 = X_1 \vee X_1 \vee \ldots X_n$	$\delta_0 \leq \sum_{i=1}^{n} \delta_i \ , \ \delta_0 \geq \delta_i$
$X_0 \leftarrow X_1 \oplus X_1 \oplus \ldots X_n$	$\delta_0 \geq \delta_i - \sum_{j \neq i} \delta_j$
$X_0 \rightarrow X_1 \oplus X_1 \oplus \ldots X_n$	$\delta_0 \leq \sum_{i=1}^{n} \delta_i \leq n + (1-n)\delta_0$
$X_0 = X_1 \oplus X_1 \oplus \ldots X_n$	$\delta_0 \leq \sum_{i=1}^{n} \delta_i \leq n + (1-n)\delta_0 \ , \ \delta_0 \geq \delta_i - \sum_{j \neq i} \delta_j$

Table 3.4: Conversion of basic logical statements into mixed-integer inequalities via extended symbolical method.

3.3.3 Geometrical Method

Geometrical method or truth table method [MoBeMi02] has two steps. First step is that the set of points in $[0,1]^n$ satisfying the Boolean function (3.2a) or Boolean formula (3.2b) is computed. Each row of the truth table is associated with a vertex of the hypercube $\{0,1\}^n$. The vertices are collected in a set V of valid points, the rest of the points $\{0,1\}^n \setminus V$ are called invalid. The valid point is a satisfying truth assignment for a Boolean formula. The mixed-integer inequalities are than obtained by computing the convex hull of V, for which several tools are available e.g. [Fuku01]. We can define the set of valid integer points as following

$$P_{CH} = \left\{ x \in [0,1]^n : x \in conv(V) \right\} \tag{3.4}$$

Where P_{CH} is a polytope defined by all the valid points of Boolean formula, and $conv(V)$ defines a convex hull of the set of valid points V. This method allows an automatic translation of truth table representing Boolean formulas into mixed-integer linear inequalities [Mig].

3.4 Big-M modeling

3.4.1 Big-M conversion

We are using so called big-M formulation as a modeling framework for modeling a complex or logic constraints and functions by converting and reformulating them into form of mixed-integer models. The idea of big-M formulation is based on forcing different constraints to be active or inactive by adding extra binary variables as "indicators of validity" for constraints into the model. By transforming of model of hybrid system into the mixed-integer inequalities we are able to capture both continuous and also discrete parts of the system into single by numerical solvers computable and feasible model.

The basic procedure for creating a big-M formulation from any constraint or a function is to decompose their descriptions into a set of *it-then-else* conditions, which are easy to model by using auxiliary binary variables. This is done by adding large positive value of constant *M* in each constraint and this value is multiplied by binary variable $\delta_i \in \{0,1\}$. Let's have a function $f_i(x)$ which should be active only if binary variable $\delta_i = 0$ and zero when binary variable $\delta_i = 1$. The corresponding big-M formulation of previous *if-then-else* condition is following

$$m(1-\delta_i) \le f_i(x) \le M(1-\delta_i) \tag{3.5}$$

Notice that even if the method is called big-M, setting the values of the constants (m, M) to be very large or even infinite can work in theory, but in practice it will cause considerable numeric drawbacks and most of the solvers will be inefficient or they don't have to find the optimal solution at all. On the other hand if the values of constants will be very small, the optimal solution in the original problem could be cut away and the problem won't be

feasible anymore. Therefore it is very important that constants (m, M) should be estimated so close as possible to lower (3.6a) and upper (3.6b) bounds of function $f_i(x)$:

$$m \cong \min f_i(x) \tag{3.6a}$$

$$M \cong \max f_i(x) \tag{3.6b}$$

Proper value of (m, M) can be determined by setting numerical constraints or pre-computing of possible values of functions used in big-M formulation. But this is not always possible for several reasons as unbounded real variables or unknowing of the behavior of functions used in big-M formulations. Therefore accurate estimation of (m, M) is a crucial task in formulation of good big-M formulations. Theoretically, an under (m) or over (M) estimate of constants suffices for our purpose. However, more realistic estimates provide computational benefits [Wil01].

3.4.2 DHA, PWA and Big-M Formulation

Big-M conversion can be understood as suitable continuous-logical modeling framework for hybrid systems represented either as PWA or DHA, what is in the interest of investigation of this thesis. As we know SAS (part of DHA) or PWA can be rewritten as the combination of linear terms and *if-then-else* rules as is shown in Chapter 2 in section Hybrid Models. Because of this nature, these models are suitable for big-M conversions into the mixed-integer inequalities by using techniques mentioned in section below called “Big-M models library”. Moreover the events (2.3) of DHA can be also expressed as big-M formulations in following form

$$f_H\left(x_r(k),u_r(k),k\right) \le M\left(1-\delta_i\right) \tag{3.7a}$$

$$f_H\left(x_r(k),u_r(k),k\right) \ge m\delta_i \tag{3.7b}$$

Conversion of logical statements typical for mode selector (MS) of DHA, can be also easily done by using techniques mentioned in section named “Propositional calculus and mixed-integer inequalities”.

3.4.3 Big-M Models Library

In this section we are representing several big-M models of most commonly used logical relations between binary variables, linear equalities and inequalities or polytopic constraints. At first let's present some basic assumptions, we denote $\delta_i \in \{0,1\}$ as a binary variable, and $x \in X \subseteq R,$ as a real variable. Linear inequality constraint is defined as $\{a^T x - b \le 0\}$, where a^T and b are scalar vectors. Similar we define also linear equality constraint as $\{a^T x - b = 0\}$. Multiple linear inequalities i.e. polytope is defined as $\{Ax - B \le 0 \leftrightarrow \delta_0 = 1\}$ (1.3), where $i = 1,\ldots,n$. And integer linear inequality is defined as $\{a_b^T x - b_b \le 0\}$, where index *b* denotes that coefficients a_b^T and b_b can obtain only integer values. And (*m, M*) are positive scalar values set as mentioned in previous section. Than the representation of corresponding big-M models are shown in following Table 3.5.

Type of statement	**Big-M model (MIP inequality)**
binary variable → linear inequality $\{\delta \to a^T x - b \le 0\}$	$a^T x - b \le M(1-\delta)$
binary variable → linear equality $\{\delta \to a^T x - b = 0\}$	$m(1-\delta) \le a^T x - b \le M(1-\delta)$
binary variable → polytope inclusion $\{\delta \to a_i^T x - b_i \le 0\}$	$a_i^T x - b_i \le M_i(1-\delta)$
linear inequality → binary variable $\{a^T x - b \le 0 \to \delta = 1\}$	$m\delta \le a^T x - b$
Integer linear inequality → binary variable $\{a_b^T x - b_b \le 0 \to \delta = 1\}$	$\left(m + \frac{1}{2}\right)\delta + \frac{1}{2} \le a^T x - b$
linear equality → binary variable (because it is impossible to do this directly, we have to use the same techniques as for polytope inclusion ↔ binary variable described below) $\{-r \le a^T x - b \le r \leftrightarrow \delta = 1\}$	$m\delta_i - r \le a_i^T x - b_i$ $a_i^T x - b_i \le r + M(1-\delta_i)$ $\delta_0 \ge 1 + \sum_{i=1}^{n} \delta_i - n$ $\delta_0 \le \delta_i, \quad r > 0$

linear inequality ↔ binary variable $\{a^T x - b \le 0 \leftrightarrow \delta = 1\}$	$m\delta \le a^T x - b$ $a^T x - b \le M(1-\delta)$
Integer linear inequality ↔ binary variable $\{a_b^T x - b_b \le 0 \leftrightarrow \delta = 1\}$	$\left(m + \frac{1}{2}\right)\delta + \frac{1}{2} \le a^T x - b$ $a^T x - b \le \frac{1}{2} + \left(M + \frac{1}{2}\right)(1-\delta)$
polytope inclusion ↔ binary variable $\{Ax - B \le 0 \leftrightarrow \delta_0 = 1\}$	$m\delta_i \le a_i^T x - b_i$ $a_i^T x - b_i \le M(1-\delta_i)$ $\delta_0 \ge 1 + \sum_{i=1}^{n} \delta_i - n$ $\delta_0 \le \delta_i$
linear inequality → linear inequality $\{a^T x - b \le 0 \rightarrow c^T x - d \le 0\}$	$m\delta \le a^T x - b$ $c^T x - d \le M(1-\delta)$

Table 3.5: Big-M models library.

Big-M formulation has proved to be very convenient modeling framework for constructing MIP problems, but there are still few numerical drawbacks, which arises not from the theory itself, but from computers nature of representation the numbers. They are lots of references about computer representation of numbers which can be found on internet, just for curiosity we recommend to the reader following sources e.g. [ASan01, wikiCoNu].

Each computer is able to work with specified numerical precision and the size of the numbers which can be processed by computer is also limited. It is impossible to work directly with numbers like infinity, or zero representation in floating point method is also not precisely defined. Therefore not all mathematical expressions are available to be represented in current numerical solvers without losing some precision and even if they are computable, the computation time needed for processing could be enormously large. In to the group of "hard to solve" mathematical expressions belongs for example equality constraints $\{a^T x - b = 0\}$, which are being transformed into two corresponding inequality constraints (one from each side) to force the expressions to be equal as it is shown in 6th row

of Table 3.5. Another example is strict inequality constraints $\{a^T x - b < 0\}$, which need some small numerical thresholds to be defined around zero value. Several of these techniques for handling the mathematical expressions to be numerically solvable by numerical solvers are incorporated in Big-M formulation, as you can see in a Table 3.5. In Big-M modeling there are also used techniques for conversion of logical expressions into MIP inequalities defined in Table 3.4. As an example you can look on 9th row of Table 3.5 where is defined Big-M representation of statement (polytope inclusion ↔ binary), where the Big-M conversion of logical AND operator from Table 3.4 has been used. The reader can found more of practical examples about Big-M formulation and mixed-integer programming in following references [YalMIP, YalBigM].

3.5 Efficient Modeling of Hybrid Systems

The complexity of resulting MIP model translated from logical propositions is significantly determined on number of binary variables included in the model. The complexity of MIP model rises exponentially with increasing number of involved binary variables, what has huge impact on speed of solving of the mathematical optimization problems. Solving of MIP optimization problems is necessary for synthesis, analysis and control of hybrid models. Therefore a reduction of a size of the model seems to be a crucial task in efficient modeling approach for these models.

In this chapter we will show that it is possible to encode and replace the binary states of DHA by fewer new auxiliary binary variables in the form of linear integer inequalities, we are calling this technique “binary encoding” of integer variables. One original binary variable will be assigned exactly with one integer inequality. Each inequality will be unique combination of valid points of truth table (geometrically vertex of a hypercube) composed for Boolean function of auxiliary variables. By this approach we can reduce the number of necessary binary variables logarithmically comparing to original MIP problem, as we will show in following pages of this thesis. Later on we are presenting also efficient technique for removing the infeasible combinations of new auxiliary variables i.e. invalid points of truth table from feasible regions of resulting MIP problem. This is done by adding extra constraints (cuts) into the MIP problem. This procedure is necessary for avoiding the model to be

declared in infeasible and undefined regions which could appear in MIP formulation, what in theoretical way should keep correct DHA trajectory and ensure well posedness of the hybrid model.

As a main reference for this Chapter we should mention work [Olaru], in which is author investigating and proposing enhanced techniques for hyperplane arrangement in MIP problems. In this paper are presented ideas for reduction the binaries on logarithmic size by "binary encoding", which we are using in our modeling framework and has been also mentioned earlier in paper [BemMor]. In paper [Olaru] there is also shown technique for constraints (cuts) reduction, based on grouping the cuts together, when a single cut is used for separation more than one infeasible combination of auxiliary variables (geometrically vertex of hypercube). By this approach we can obtain significant decrease of number of used cuts. Moreover with small changes in cuts creation (by using so called deep cuts) we improved this technique of "reduced cuts" to technique of "reduced deep cuts", which are constructing more suitable representation of MIP model for current numerical solvers. We are presenting this technique later on in this chapter.

3.5.1 Binary Encoding of Integer State Variables

Here we will present the general technique for binary encoding of original Boolean variables $X_i \in \{0,1\}^{n_x}$ by logarithmic number of new auxiliary binary variables $\delta_j \in \{0,1\}^{n_d}$ with using the Truth Table Method (Geometrical Method) defined few sections above. This technique is based on defining the original binary variable as a Boolean function $(3.2a)$ of new auxiliary binary variables, where each binary variable X_i is associated with one corresponding row of the truth table, geometrically represented as a vertex of hypercube. Each row of the table is represented as a unique combination of new variables δ_j or it can be conceived also as a unique Boolean formula of variables δ_j. The maximum number of all possible combinations of variables δ_j equals 2^{n_d} where n_d means cardinality of new auxiliary binary variables. Hence the number of new auxiliary binary variables n_d needed for encoding the original binary variables in cardinality of n_x will be set by relation (3.8), where the brackets $\lceil . \rceil$ denotes the ceiling operator.

$$n_d = \lceil \log_2(n_x) \rceil \tag{3.8}$$

As mentioned before in section about Geometrical Method, the truth table can be set up e.g. by enumeration of corresponding Boolean formulas. More detailed definition of a truth table can be also found in work [Mig] or on a web [wikiTT].

For demonstration we will create a table of all possible combinations of three auxiliary binary variables $\{\delta_1, \delta_2, \delta_3\}$. Where each row of the table can be represented as a different Boolean formula in CNF, therefore we can use a transformation techniques proposed in the Table 3.4 to transform these Boolean expressions into equivalent mixed-integer linear inequalities, as is shown in following example 3.2. The $\leftrightarrow$ operator means that all transformation procedures are equivalent and reverse.

Example 3.2:

Truth Table				**Boolean formula**		**Mixed-integer inequality**
δ_1	δ_2	δ_3				
0	0	0	$\leftrightarrow$	$\neg\delta_1 \wedge \neg\delta_2 \wedge \neg\delta_3$	$\leftrightarrow$	$-\delta_1 - \delta_2 - \delta_3 \geq 0$
0	0	1	$\leftrightarrow$	$\neg\delta_1 \wedge \neg\delta_2 \wedge \delta_3$	$\leftrightarrow$	$-\delta_1 - \delta_2 + \delta_3 \geq 1$
0	1	0	$\leftrightarrow$	$\neg\delta_1 \wedge \delta_2 \wedge \neg\delta_3$	$\leftrightarrow$	$-\delta_1 + \delta_2 - \delta_3 \geq 1$
0	1	1	$\leftrightarrow$	$\neg\delta_1 \wedge \delta_2 \wedge \delta_3$	$\leftrightarrow$	$-\delta_1 + \delta_2 + \delta_3 \geq 2$
1	0	0	$\leftrightarrow$	$\delta_1 \wedge \neg\delta_2 \wedge \neg\delta_3$	$\leftrightarrow$	$\delta_1 - \delta_2 - \delta_3 \geq 1$
1	0	1	$\leftrightarrow$	$\delta_1 \wedge \neg\delta_2 \wedge \delta_3$	$\leftrightarrow$	$\delta_1 - \delta_2 + \delta_3 \geq 2$
1	1	0	$\leftrightarrow$	$\delta_1 \wedge \delta_2 \wedge \neg\delta_3$	$\leftrightarrow$	$\delta_1 + \delta_2 - \delta_3 \geq 2$
1	1	1	$\leftrightarrow$	$\delta_1 \wedge \delta_2 \wedge \delta_3$	$\leftrightarrow$	$\delta_1 + \delta_2 + \delta_3 \geq 3$

Table 3.6: Example for conversion of truth table into equivalent Boolean formulas and mixed-integer inequalities.

As we can see in previous example for each row of the truth table there is exactly one corresponding mixed-integer linear inequality, with which we can replace an original binary variable X_i in MIP problem. It is obvious that in this example the number of binaries X_i what we can encode with this approach by three auxiliary binary variables $\{\delta_1, \delta_2, \delta_3\}$ is set

by relation 2^{n_d} defined before, hence equals $2^3 = 8$. On figure 3.1 is show the corresponding visual representation of hypercube $\{0,1\}^3$ for truth table from example 3.2.

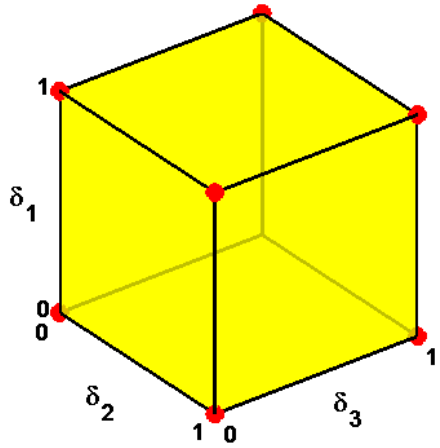

Figure 3.1: Visual representation of truth table from example 3.2 as a 3-dimensional hypercube with valid points $\{0,1\}^3$.

Technique of "binary encoding" of integer variables allows us to significantly decrease the number of binaries involved in our model what goes hand in hand with decreasing of complexity of whole model, but nothing is for free and several drawbacks appears also in this approach.

First cost what we have to pay is hidden behind new Big-M conversions which need to be done. In "binary encoding" approach a single binary variable is replaced by corresponding mixed-integer linear inequality, because of this the logical implications and equivalences used for modeling are becoming more complex and therefore more difficult to transform into Big-M formulations as it was with using only single binary variable. For this purposes there are available efficient transformation techniques for more complex statements (e.g. integer linear inequality ↔ binary variable) as is shown in Table 3.5 which is containing the Big-M models library. So even if we reduce the number of binary variables what decreases the complexity of the model, the number of MIP inequalities needed for description of the model is rising, due to more complex Big-M formulations, what has certainly negative effect on complexity of the model created by this approach. Or sometimes even a new auxiliary binary variable which is used for description of a complex statement is needed to be incorporated into the model, what is also increasing the complexity of the model. Here we

have to be careful and compare the pros (decreased number of binaries) and cons (increased number of MIP constraints and auxiliary variables) of this approach for particular model situation and its effect on complexity of resulting model.

Second problem what we have to deal with appears when the number of original binary variables X_i what we want to encode is not equal to powers of two 2^{n_d}. As we mentioned before the number of new auxiliary variables needed for encoding n_x states is equal to the number set by relation (3.8), so the number of auxiliary variables and their descriptive power is not arbitrary choice. In this approach we may face a situation when there will be more possible combinations of the auxiliary variables (tuples or nodes) than we actually need for encoding of original binary variables. And here the problem of infeasible tuples of auxiliary binary variables or unallocated vertexes of hypercube arises. Note that the number of unallocated tuples n_u may obtain a significant value especially with higher number n_x of original binary variables. The number n_u is function of n_x and is set by following relation (3.9).

$$n_u = 2^{\lceil \log_2(n_x) \rceil} - n_x \tag{3.9}$$

We will demonstrate this problem on following example 3.3.

Example 3.3: Let us consider a model with five binary states X_i , where $i = 1,\ldots,5$. These can be represented by three auxiliary binaries $\{\delta_1, \delta_2, \delta_3\}$, for whose the truth table was shown in example 3.2. To each row of the truth table and to corresponding MIP inequality we will assign exactly one binary state X_i. Note that there are three rows of the table left without any assignment, therefore these three tuples are unallocated. The number $n_u = 3$ fits also with equation (3.9), because in this example $n_d = 3$ and $n_x = 5$. But this is something what a numerical solver can't realize without our help. The solver will consider these unallocated nodes as a feasible solutions (because we didn't say to him opposite), and could lead the optimization into undefined regions of MIP problem, what will cause the crash of our model.

On figure 3.2 is shown the visual representation of hypercube for the example 3.3, with three unallocated nodes marked with blue crosses. These unallocated nodes are representing last three rows of truth table from example 3.2.

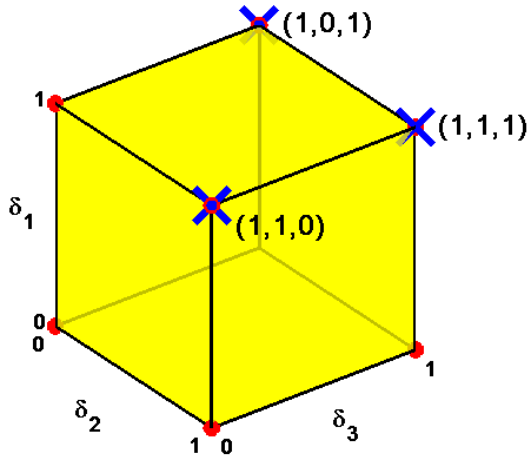

Figure 3.2: Visual representation of hypercube from example 3.3 with three unallocated tuples $\{(1,0,1);\ (1,1,0);\ (1,1,1)\}$.

We need to somehow say to the solver that this tuples where no actual binary state is defined should be put out of feasible regions of constructed MIP problem. The solution of this problem will be shown in following stage about cuts, where the extra linear constraints will be added into the model for eliminating these infeasible combinations of auxiliary variables.

3.5.2 CUTS

By the notion of "cuts" we are meaning extra constraints added to the model to restrict infeasible integer or possible non-integer solutions that would be solutions of the continuous relaxation of MIP problem computed by most of the solvers. By this approach we can ensure that we will find optimal solution and significantly reduce the number of branches needed to solve MIP problem and consequently improve the speed of obtained solution. Therefore the demand for finding a "good" cuts and incorporating them into the model plays very important role in efficient modeling of MIP problems.

In this part of the thesis we are investigating several approaches for removing infeasible nodes (unallocated vertexes of hypercube) from feasible regions of MIP problem by cuts.

Visual demonstrations of used techniques will be shown on 3-D hypercube from example 3.3 constructed as a combination of three binary variables $\{\delta_1, \delta_2, \delta_3\}$.

3.5.2.1 One node one cut approach

This is a basic approach to solve the problem with unallocated nodes (infeasible integer solutions). Main idea is to use exactly one mixed integer inequality for cutting away one particular infeasible node from a model, as can be seen also on figure 3.3. This approach is very easy to implement, because the representing mixed integer inequalities for a single node can be found by using transformation techniques shown in Table 3.4. For this purposes we also present following definition.

Definition 3.1: us consider a collections of all possible combinations of n binary variables $\{\delta_1, \ldots, \delta_n\}_k$ with cardinality $k = 1, \ldots, 2^n$, which are shaping an n-dimensional hypercube and each of them can be conceived as a vertex of this hypercube. Then we can find a constraint (3.10) for each vertex, which is putting this particular combination of binaries and only this combination infeasible.

$$-\sum_{i \neq j}^{n_T} \delta_i - \sum_{j \neq i}^{n_F} (1 - \delta_j) \leq -\varepsilon \qquad (3.10)$$

Note that n_T represents the number of binaries in tuple (node), which value is equal to 1 or TRUE, and n_F represents the number of binary variables with 0 or FALSE value. Then the total number of binaries in one tuple must hold following equality $n_T + n_F = n$. Geometrical representation of each constraint of type (3.10) is a hyperplane which is separating the space in to two half spaces, one feasible and one infeasible.

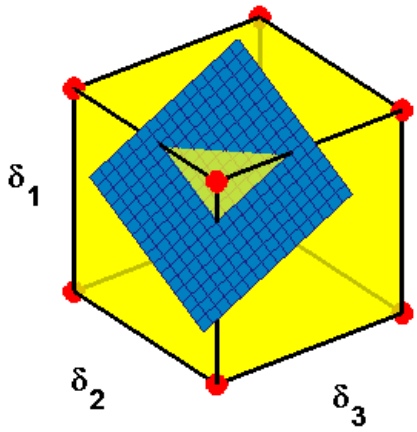

Figure 3.3: A single cut (hyperplane - blue) separating one vertex of the hypercube.

Example 3.4: We will demonstrate this basic approach on three infeasible nodes from Example 3.3, with using three single cuts to restrict them from feasible region of the model. Corresponding MIP inequalities (3.11) for restricting these nodes are shown in following Table 3.7.

Infeasible node $\{\delta_1,\delta_2,\delta_3\}$	**Corresponding cut (MIP inequality)**	
$(1,0,1)$	$\delta_1-\delta_2+\delta_3\le 2-\varepsilon$	(3.11a)
$(1,1,0)$	$\delta_1+\delta_2-\delta_3\le 2-\varepsilon$	(3.11b)
$(1,1,1)$	$\delta_1+\delta_2+\delta_3\le 3-\varepsilon$	(3.11c)

Table 3.7: MIP inequalities for restricting infeasible nodes $\{(1,0,1);\ (1,1,0);\ (1,1,1)\}$.

Note that there is extra scalar value $\varepsilon\in(0,1)$ deducted from right side of the MIP inequality, this value says how "deep" the cuts will be and how much space from the feasible region they will cut. So when the scalar $\varepsilon=0$, it means that this cut will be very strict and it will lie exactly on node that should separate, on the other hand when $\varepsilon=1$, it means that this cut is deep and is cutting away from feasible region as much space as possible. Although the choice of the value $\varepsilon\in(0,1)$ is arbitrary, it has a huge impact on model efficiency during the optimization as will be demonstrated later in section about deep cuts approach.

We used value $\varepsilon=0.5$ in our example 3.4 just for a demonstration of above mentioned issues, than the corresponding visual representation of these cuts is shown on figure 3.4. On

the left picture there are shown three restricted areas of hypercube defined by cuts, which are containing the infeasible nodes, these infeasible regions are painted by blue color and the remaining feasible region is painted yellow. On the right picture, there is shown a polytopic representation of the feasible region for corresponding MIP model after executing the cuts. Where red nodes are representing feasible integer solutions and blue nodes are representing non-integer solutions, which are naturally infeasible for MIP problem.

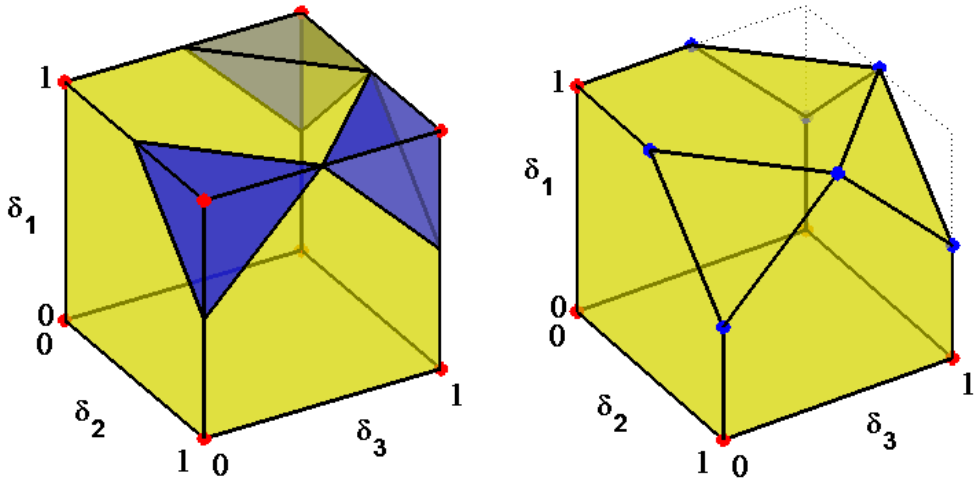

Figure 3.4: *Left* - Single cut approach for cutting away three nodes by cuts (3.11) from example 3.4. *Right* – remaining feasible region with 7 non-integer nodes (blue).

Disadvantage of this approach is that for more complex problems with a lot of unallocated nodes we need to incorporate a huge number of extra constraints to the model, what is making the model even more complex and less efficient for solving optimization problems. The efficient technique for minimizing the number of constraints can be found in paper [Olaru] and we are describing it in following section named “Reduced cuts approach”.

Second even bigger drawback lies in a polytopic representation of feasible regions for resulting MIP model. As we can see on right picture from figure 3.4 after adding the cuts in to the model, new non-integer nodes (blue) appears in geometrical representation of feasible regions of the model. These nodes obviously can’t be the solutions of the MIP problem, but for continuous relaxations of MIP problem these nodes are representing feasible solutions of relaxed LP problem. So for all the current solvers based on algorithms using LP relaxations of the MIP problem would be this representation of the MIP model

highly inconvenient and inefficient, due to increased number of branches needed to be checked, or for some advanced solvers there would be need for presenting their own cuts for restricting these infeasible solutions from continuous relaxations. Therefore for further use of the model is very important to provide "good" cuts, which will minimize the number of non-integer nodes in polytopic representation of feasible regions of the MIP model, what will provide significant relief in number of the LP relaxations of MIP problem needed to be solved by numerical solvers. We will focus on this issue in following pages of this thesis in section named "Reduced deep cuts", where we are trying to combine the technique of reduced cuts and technique of deep cuts to provide more efficient representations of MIP models for current numerical solvers.

3.5.2.2 Reduced cuts approach

By this approach we are able to significantly reduce the number of cuts incorporated in the model. This is done by merging single cuts for particular nodes together to create one complex cut for cutting away more than one node.

The idea is based on finding and grouping the neighbor nodes of hypercube together and introducing a single inequality for their restriction from feasible region. By meaning of neighbor node we mean for example two vertexes which are forming an edge of hypercube, or four vertexes of hypercube which are forming one plane of a hypercube. So generally we can group together and separate by single cut 2^n nodes, which are forming n-dimensional hyperplane which is part of a multi-dimensional hypercube representing corresponding combinations of used binary variables.

Now arise the question what can be the best possible allocation of infeasible nodes in the hypercube, for grouping as much as possible neighbor nodes together. Even if the allocation of nodes is arbitrary and one can choose different combinations, we are proposing following ordering of binary variables to be most favorable for reduced cuts approach, because of the maximum number of neighbor nodes placed next to each other. The idea is to use basic, most simple ordering of nodes where we are gradually changing the values of binaries in descending order from the last binary to first binary one by one, what geometrically means that we are changing the position from one vertex to another vertex by moving on edges of the hypercube. You can check that this is true e.g. by enumeration of vertexes of hypercube

from figure 3.1. We are using this ordering in every example of this thesis and can be seen e.g. in truth table for example 3.2. With using basic ordering the number of new reduced cuts n_r depends on number of unallocated nodes (3.9) and can be approximately estimated by following relation (3.12).

$$n_r \approx \lceil \log_2(n_u) \rceil \tag{3.12}$$

Note that equation (3.12) is only raw estimation and not a strict relation. Real relation between number of reduced and number of original cuts, can be seen in Appendix on figure A.1. Moreover dependence of maximum number of original and reduced cuts on number of auxiliary binary variables δ is shown in Appendix on figure A.2.

For better understanding of this technique there are following examples 3.5 and 3.6. In example 3.5 we are presenting also figures 3.5 and 3.6, with visual demonstrations of complex cuts for vertexes of 3-dimensional hypercube. And on example 3.6 we are demonstrating the reducing constraints power of this approach.

Example 3.5: Let us consider a hypercube representation for all possible combinations of three auxiliary binary variables $\{\delta_1, \delta_2, \delta_3\}$, while three of the nodes are infeasible, as shown on Example 3.3. Separation of the infeasible nodes by single cut approach was demonstrated on Example 3.4. Now we will show that it is possible to use instead of three MIP inequalities (3.11) only two MIP inequalities (3.13) for all three nodes, to restrict them from feasible region. First what we have to realize is which of these nodes lies together an edge of the hypercube, to be able for us separate them by single cut. On figure 3.5 is clearly visible that the following pairs of nodes $\{(1,0,1);(1,1,1)\}$ and $\{(1,1,0);(1,1,1)\}$ forms together two edges of the hypercube. Than the corresponding MIP inequalities (3.13) for these edges are defined in following Table 3.8. Their visual representation is shown on figure 3.5 where on the left picture there is hypercube with cut defined by inequality $(3.13b)$ and on the right picture is shown a hypercube with cut defined by inequality $(3.13a)$.

Pair of infeasible nodes $\{\delta_1, \delta_2, \delta_3\}$	**Corresponding complex cut (MIP inequality)**	
$(1,0,1);(1,1,1)$	$\delta_1 + \delta_3 \leq 2 - \varepsilon$	$(3.13a)$

$(1,1,0);(1,1,1)$	$\delta_1 + \delta_2 \leq 2 - \varepsilon$	(3.13b)

Table 3.8: MIP inequalities for restricting two infeasible edges of hypercube.

For cuts (3.13) demonstrated on figures 3.5 and 3.6 we also used value $\varepsilon = 0.5$ as in example 3.4, this was done for a better comparison of these two techniques. Moreover on figure 3.6 is shown a remaining feasible region after executing these complex cuts (3.13) with new 6 non-integer nodes.

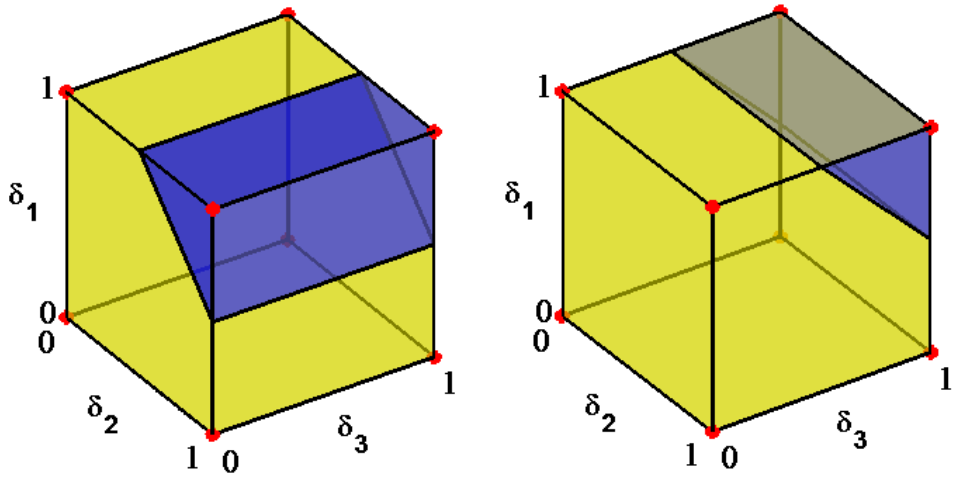

Figure 3.5: Complex cuts for edges of the hypercube. *Left* – cut for two nodes defined by equation $(3.13b)$ with value $\varepsilon = 0.5$. *Right* – cut for two nodes defined by equation $(3.13a)$ with value $\varepsilon = 0.5$.

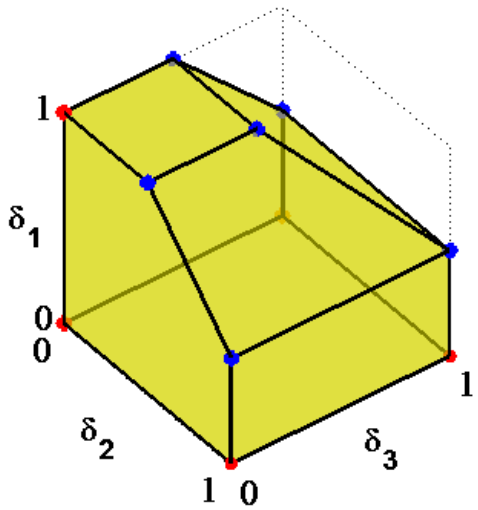

Figure 3.6: Remaining feasible region after two complex cuts with 6 non-integer nodes (blue).

Example 3.6: Assume that we have model with 9 binary states X_i, $i = 1,\ldots,9$, and we want to encode them by using 4 auxiliary binary variables $\{\delta_1, \delta_2, \delta_3, \delta_4\}$. As we know from section about binary encoding, 4 extra binaries are providing us space for encoding 2^4=16 binary states of the model. Hence after constructing the truth table and associating for each binary state X_i exactly one row of the table, there will be 7 unallocated nodes left in 4-D hypercube representation. Unfortunately we can't visualize this example, but corresponding unallocated combinations of δ_j variables and their corresponding cuts (3.14) using basic "one node one cut" approach, what is shown in following table 3.9.

Infeasible node $\{\delta_1, \delta_2, \delta_3, \delta_4\}$	**Corresponding cut (MIP inequality)**	
(1,0,0,1)	$\delta_1 - \delta_2 - \delta_3 + \delta_4 \leq 2 - \varepsilon$	(3.14a)
(1,0,1,0)	$\delta_1 - \delta_2 + \delta_3 - \delta_4 \leq 2 - \varepsilon$	(3.14b)
(1,0,1,1)	$\delta_1 - \delta_2 + \delta_3 + \delta_4 \leq 3 - \varepsilon$	(3.14c)
(1,1,0,0)	$\delta_1 + \delta_2 - \delta_3 - \delta_4 \leq 2 - \varepsilon$	(3.14d)
(1,1,0,1)	$\delta_1 + \delta_2 - \delta_3 + \delta_4 \leq 3 - \varepsilon$	(3.14e)
(1,1,1,0)	$\delta_1 + \delta_2 + \delta_3 - \delta_4 \leq 3 - \varepsilon$	(3.14f)
(1,1,1,1)	$\delta_1 + \delta_2 + \delta_3 + \delta_4 \leq 4 - \varepsilon$	(3.14g)

Table 3.9: Example of basic approach with 7 single cuts for 7 infeasible nodes.

Now we apply the results of this section to reduce the number of cuts by merging the neighbor nodes together and separate them by single cut. In a table 3.10 we can see that for this particular case we were able to replace 7 single cuts (3.14) constructed by basic approach just with 3 complex cuts (3.15) constructed by using "reduced cuts" approach. Note that the group of 4 infeasible nodes $(3.15a)$ is forming one plane of a hypercube and the pairs of infeasible nodes $(3.15b, 3.15c)$ are forming two edges of a hypercube. The values of particular binaries highlighted bold are representing common values for the group of nodes, from which the corresponding MIP inequalities were derived.

Group of infeasible nodes $\{\delta_1,\delta_2,\delta_3,\delta_4\}$	**Corresponding cut (MIP inequality)**	
(**1,1,**0,0); (**1,1,**0,1); (**1,1,**1,0); (**1,1,**1,1)	$\delta_1+\delta_2\le 2-\varepsilon$	(3.15a)
(**1,0,1**,0); (**1,0,1**,1)	$\delta_1-\delta_2+\delta_3\le 2-\varepsilon$	(3.15b)
(**1,0,**0,**1**); (**1,0,**1,**1**)	$\delta_1-\delta_2+\delta_4\le 2-\varepsilon$	(3.15c)

Table 3.10: Example of 3 complex cuts for restricting 7 infeasible nodes.

As shown on example 3.6 this approach provides great improvement in number of constraints what we have to incorporate into the model, what is decreasing the complexity of the resulting model. But on the other hand as we can see on example 3.5 exactly on figure 3.6 the number of non-integer vertexes which are created by using this approach is still considerable large what has negative effect on computing efficiency of the model. We will show how to easily handle this drawback in next section of this thesis, by providing deep cuts.

3.5.2.3 Deep cuts approach

In this section we want to prove that reduced cuts approach is obtaining best results with performing so called deep cuts i.e. with value of $\varepsilon=1$ for cuts defined by equation (3.10). The cut is called deep because is restricting maximum possible space from feasible region while still holding the condition for cutting away desired nodes.

Using of this approach can be seen on figures 3.7 and 3.8 where we used cuts (3.13) from example 3.5, the only difference here is that we changed the value of ε from 0.5 to 1. Than we can compare the figures 3.6 and 3.8 with remaining feasible regions after executing cuts of these two approaches. As is clear from this comparison, with performing deep cuts we are obviously creating more suitable model for mathematical optimization, because in model demonstrated by figure 3.8 there are no extra non-integer nodes, which would otherwise slow down the numerical solvers based on LP relaxations of MIP problem.

Even if we didn't propose the proof that this approach prevents from creating new non-integer vertexes in remaining polytopic representation of feasible region, we assume that the number of them is minimal in comparison with different approaches which were mentioned in this thesis.

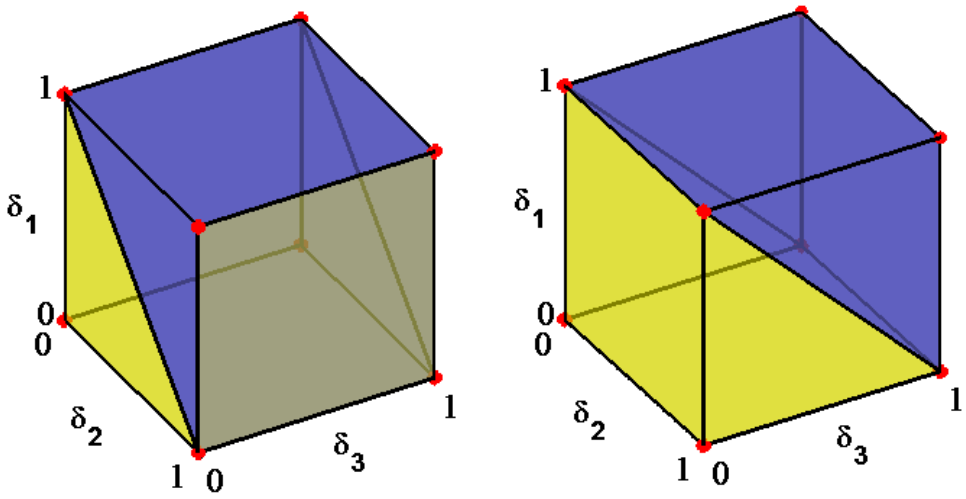

Figure 3.7: Complex deep cuts for edges of the hypercube. *Left* – deep cut for two nodes defined by equation $(3.13b)$ with value $\varepsilon = 1$. *Right* – deep cut for two nodes defined by equation $(3.13a)$ with value $\varepsilon = 1$.

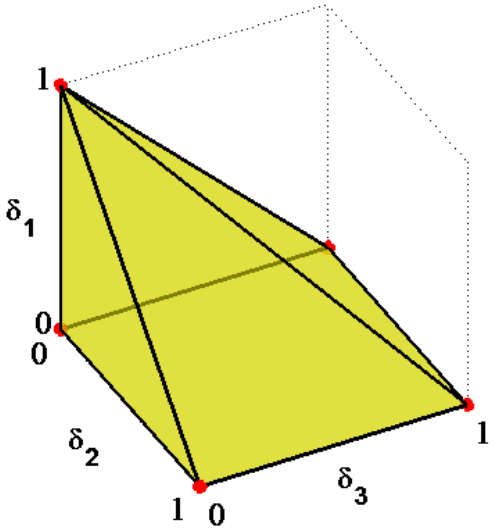

Figure 3.8: Remaining feasible region after performing two complex deep cuts with no non-integer nodes.

Chapter 4

4 Software Tools for Hybrid Modeling

In this chapter we will shortly introduce modeling tools used in this thesis, particularly modeling languages HYSDEL [Hys] and YALMIP [YAL]. Both tools are shipped with free Multi-Parametric Toolbox (MPT) for MATLAB [KGBC], designed for analysis and deployment of optimal controllers for constrained linear and hybrid systems.

In this thesis we are using several MLD models developed in HYSDEL for comparison with hybrid models created in YALMIP, what will be shown in following pages.

4.1 HYSDEL

HYSDEL (HYbrid Systems DEscription Language) is a high-level modeling language, which allows describing hybrid dynamics in textual form, than the related compiler is used for translation of textual form into computational mathematical models of hybrid systems, particularly into MLD or PWA form. As we demonstrated in chapter 2, system described in MLD form can be instantly used for optimization to solve e.g. verification or optimal control synthesis problems. For further reading and better understanding of HYSDEL modeling language, we highly recommend to the reader following sources [Hys, ToBe, Tor, BemDHS, Mig].

4.2 YALMIP

YALMIP is a powerful modeling language for advanced modeling and solution of convex and nonconvex optimization problems. It is implemented as a free (GNU license) MATLAB toolbox with rapid algorithm development. The language is in accordance with standard MATLAB syntax, what makes it extremely user friendly for common MATLAB users. The tool was initially developed in 2001 and over the years has grown enormously and today supports a broad range of optimization problems. Also a large number of various modeling

tricks are included, what helps user to focus on high-level modeling, while low-level modeling is done internally by YALMIP providing as efficient models as possible. Supported optimization classes are wide, such as linear, quadratic, second order cone, semidefinite, mixed integer conic, geometric, local and global, polynomial, multiparametric, bilevel and robust programming [YAL].

One of the central ideas of YALMIP is that it relies on external solvers carrying computation aspects, while the tool itself is focusing on the language and high-level algorithms. Complete list of supported external solvers can be found on YALMIP home page [YAL] in section "solvers". Moreover YALMIP is also shipped with internal solvers for global optimization, mixed integer programming, multiparametric programming, sum-of-squares programming and robust optimization [YAL].

For purposes of hybrid modeling we will naturally focus on construction of mixed integer models in YALMIP. We highlight the reader on chapter 3 where most important YALMIP modeling features for MIP problems are presented, especially we refer to Big-M modeling approach as a main tool used internally in YALMIP for construction of MIP problems.

4.2.1 YALMIP Hybrid Modeling Framework

Main goal of this thesis lies in creating user friendly and efficient computational modeling framework suitable for creating corresponding MIP models of hybrid systems defined as DHA or PWA. The idea was driven by exploit capabilities of YALMIP and creating MIP representations of hybrid systems with using YALMIP modeling features described in chapter 3. Motivation for creating such framework lied in that all current computational hybrid modeling tools require knowledge of specific language syntax e.g. HYSDEL. In contrast our YALMIP hybrid modeling framework is completely based on YALMIP syntax, which as mentioned before is consistent with standard MATLAB syntax, what makes modeling process in this tool very simple and intuitive for all MATLAB user. We are demonstrating "user friendliness" of framework syntax on following pages of this thesis, and efficiency of resulting models in chapter 5 as main powers of this modeling tool. Further in the text we are denoting this new framework YALMIP-FSM (YALMIP - finite state machine).

4.2.1.1 YALMIP-FSM Modeling Language Syntax

Modeling syntax consists from creating three basic input objects: *variables*, *states*, and *transitions*.

Variables object is containing information about variables names, sizes, *binary-real* indicators (declaring variable to be binary or real), and variables bounds defined in parameters section. *State* object is containing declaration of dynamical behavior for particular state. *Transition* object is carrying information about switching condition between two states and pointers on interconnected states, first for original state and second for destination state. These three objects (*variables*, *states*, *transitions*) together with declared *time horizon* and *options* are set as inputs for core function of modeling framework, which is automatically creating resulting hybrid model as a set of MIP constraints.

We will demonstrate framework's syntax simplicity on following examples 4.1 and 4.2.

Example 4.1: Let us consider a hybrid model of thermostat with 2 discrete states $(4.1\mathrm{b})$ involving different dynamical behavior. Dynamics in states are defined by difference equations: $x_{k+1} = x_k - heat$ for state1 representing cooling, and $x_{k+1} = x_k + heat$ for state2 representing heating, where x_k is continuous state variable $(4.1\mathrm{a})$ and denotes the temperature, note that the *k*-th index denotes a time step for continuous state variables. Model contains also two transitions $(4.1\mathrm{c})$ with thresholds for switching the states. Transition from state1 to state2 is defined by lower bound (LB) for temperature $x_k < 21\,^{\circ}\mathrm{C}$, and transition from state2 to state1 is set by upper bound (UB) for temperature $x_k > 23\,^{\circ}\mathrm{C}$, for numerical tightness we are using in modeling case $(4.1\mathrm{c})$ tolerance of $0.1\,^{\circ}\mathrm{C}$. Block diagram for this model is shown on figure 4.1, and moreover for demonstration of functionality an open loop simulation on time horizon *N = 15* with initial conditions $x_0 = 21\,^{\circ}\mathrm{C}$ is shown on figure 4.2.

Complete corresponding YALMIP-FSM framework syntax is demonstrated as follows (4.1).

Declaration of Variables and Parameters: (4.1a)

```
% x - temperature  u - heating switch
names = {'x' 'u' };
varsizes = {[1,1] [1,1] };
indicators = { 'r' 'b' };
% variable parameters - minimal and maximal temperature allowed
param.ineq = { '[ min_temp <= x{k} <= max_temp ]' };
param.eq = {};
param.val = { 'max_temp = 50; min_temp = 0; heat = 1;' };
% variables input object
variables = yalmip_fsm_variables(names, varsizes, indicators, param );
```

States: (4.1b)

```
% state 1 - heating off
s1 = yalmip_fsm_state('[x{k+1} == x{k} - heat ]');
% state 2 - heating on
s2 = yalmip_fsm_state('[x{k+1} == x{k} + heat ]');
states = [ s1, s2 ];
```

Transitions: (4.1c)

```
% transition 12 - jump from state 1 to state 2
t12 = yalmip_fsm_transition(s1, s2, '[ x{k} < 21.1 ]');
% transition 21 - jump from state 2 to state 1
t21 = yalmip_fsm_transition(s2, s1, '[ x{k} > 22.9 ]');
trans = [t12, t21];
```

Core Function and Options: (4.1d)

```
% 'unary' - basic approach; 'binary' - binary encoded states
Options.encoding = 'unary';
% 'basic' - one node one cut; 'enhanced' - reduced cuts
Options.cuts = 'basic';
% 0 - silent, 1 - elapsed time
Options.verbose = 1;
% core modeling function, fsm - outgoing model, V - internal variables
[fsm, V] = yalmip_fsm(states, trans , variables, N , Options)
```

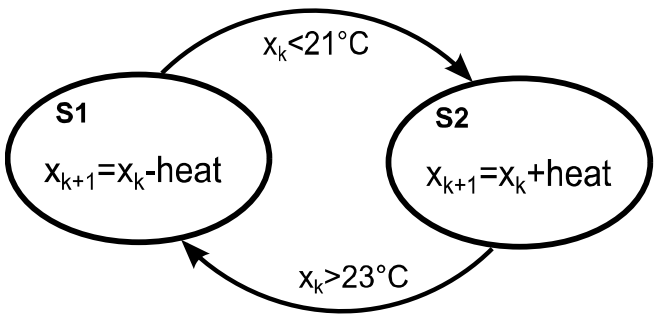

Figure 4.1: Block diagram of hybrid model of a thermostat from example 4.1.

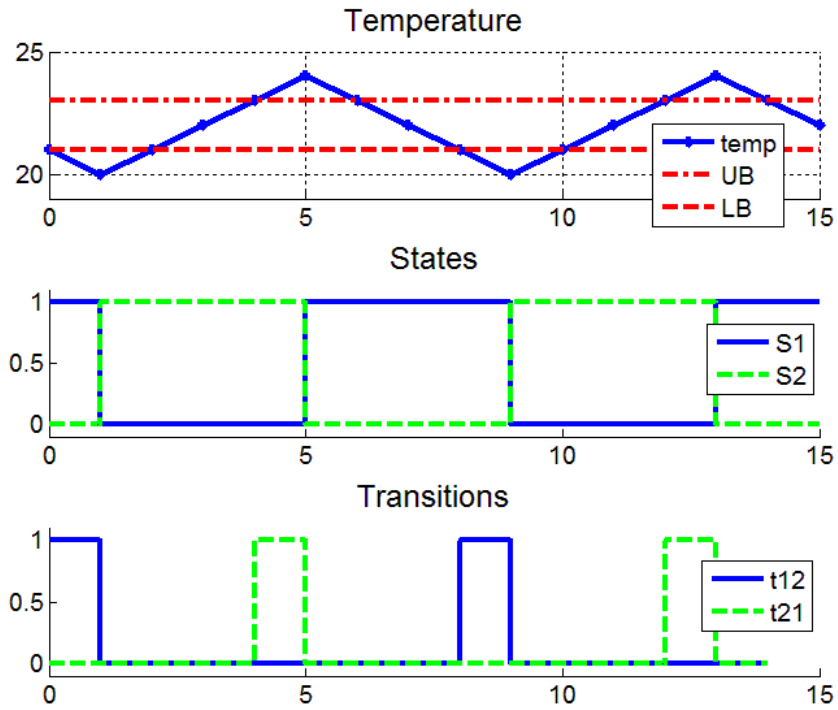

Figure 4.2: Open loop simulation of hybrid model from example 4.1.

Example 4.2: In this example we assume to have hybrid model of a truck with 4 discrete states. First state *S1* represents a loading of a truck in static loading place (i.e. not moving original destination: $D_k \leq 0$). Loading of a truck is represented by difference equation $C_{k+1} = C_k + load_k$, where C_k represent capacity (how much load can truck carry) of a truck in *k*-th time step. State *S2* represents travelling of a truck toward target destination what is defined by difference equation $D_{k+1} = D_k + speed_k$, where D_k denotes a distance of a truck from origin destination in *k*-th time step. State *S3* represents unloading of a truck in static unloading place (i.e. not moving target destination: $D_k \geq 300$). Unloading of a truck is represented by differential equation $C_{k+1} = C_k - unload_k$. State *S4* represents travelling of a truck back to original destination what is defined by equation $D_{k+1} = D_k - speed_k$. Switching conditions for "static" states *S1* and *S3* to "traveling" states *S2* and *S4* are represented by binary variable $move_k$, while switching conditions from states *S2* and *S3* and from state *S4* to *S1* are defined by distance thresholds $D_k \geq 300$ and $D_k \leq 0$.

Consistent description of the states and switching conditions for this model is demonstrated on block diagram shown on figure 4.3. Corresponding YALMIP-FSM code, with defined

variables and parameters can be found in Appendix (code A.1). Open loop simulation on time horizon *N = 15* with initial conditions (*S1=1; $load_0$=0; $unload_0$=0; $distance_0$=0; $capacity_0$ =0*) is shown on figures 4.4 and 4.5.

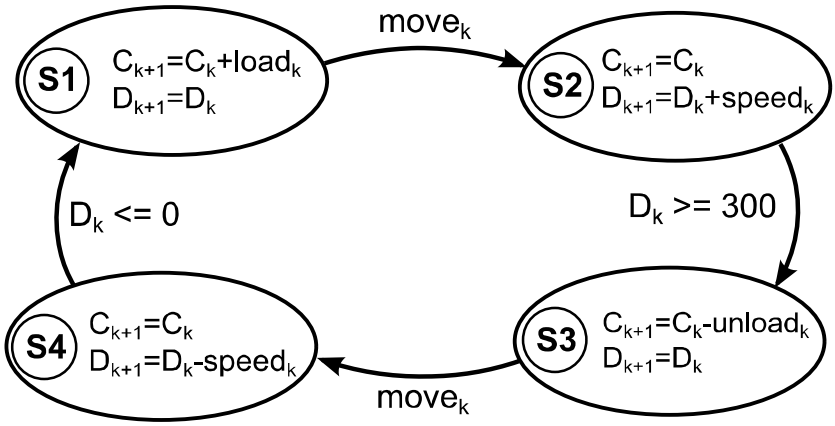

Figure 4.3: Block diagram of hybrid model of a truck from example 4.2.

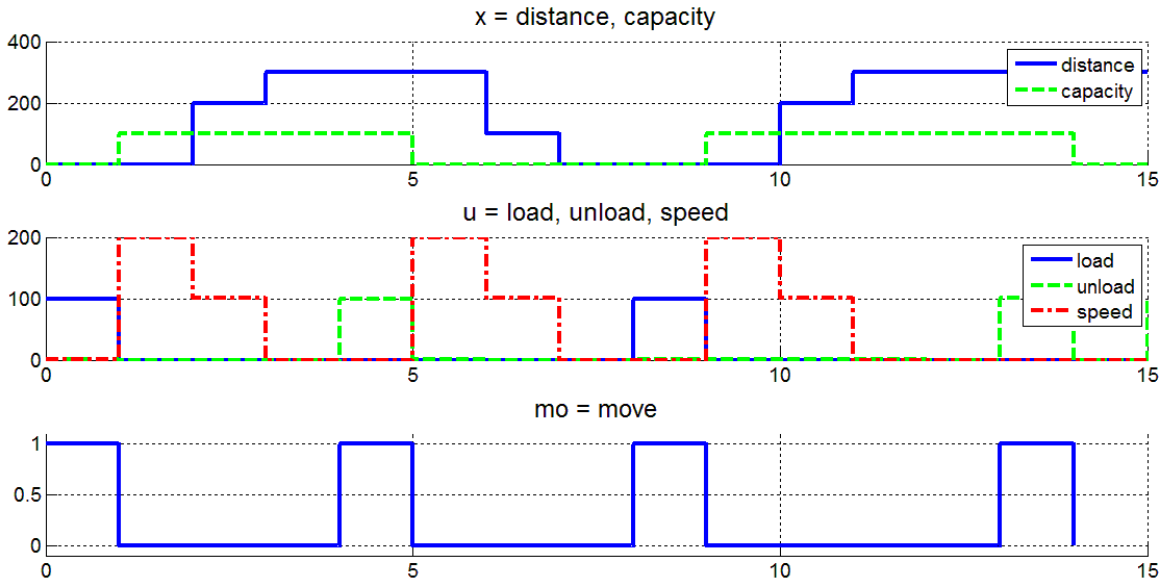

Figure 4.4: Continuous states and inputs behavior in open loop simulation of hybrid model of a truck from example 4.2.

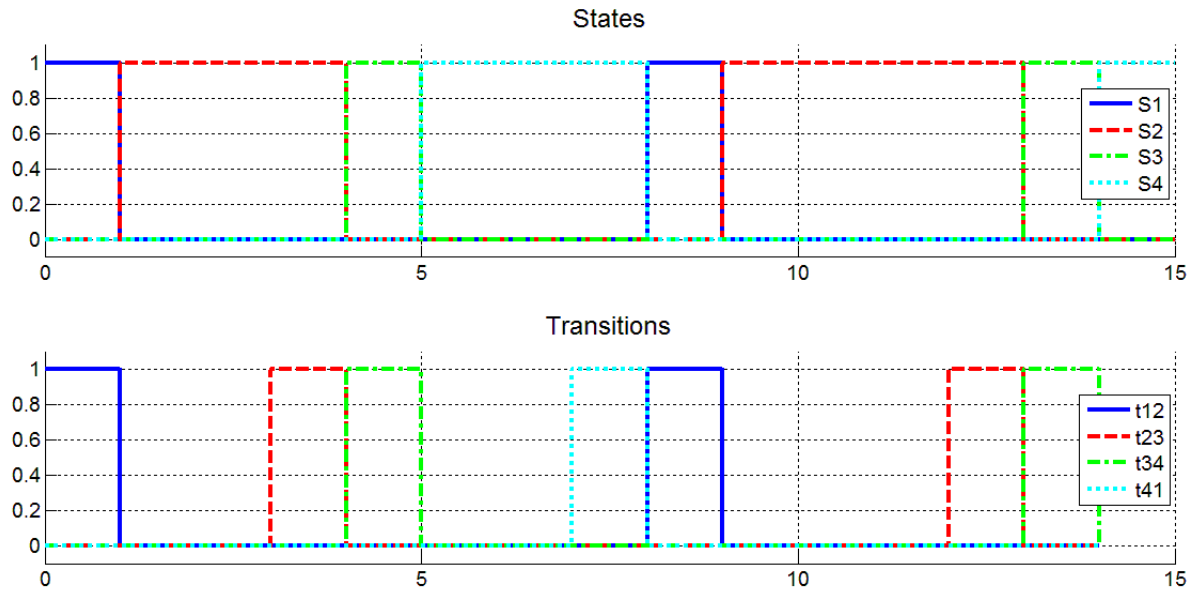

Figure 4.5: States (S) and transitions (t) behavior in open loop simulation of hybrid model of a truck from example 4.2.

4.3 Computational Aspects

In this section we computationally test the hybrid models created in HYSDEL and in YALMIP-FSM framework. For this test we used hybrid model of a truck with 4 binary states defined in previous section example 4.2, corresponding codes for YALMIP-FSM models (code A.1) and HYSDEL models (code A.2, code A.3) are enclosed in Appendix. All models were solved using commercial solver Gurobi 4.6.1 on Intel® Core™ i3 CPU 2.40 GHz 64-bit Windows 7 workstation with 3GB of RAM.

In our test we used four different models, two models with different modeling approaches defined in chapter 3, for each modeling framework (HYSDEL, YALMIP-FSM). First we used basic approach with unary encoding of state binary variables, what means that for each binary state we used exactly one binary variable in model, second used approach was binary encoding approach of state variables, what means that for each binary variable we allocated exactly one unique combination of auxiliary binary variables. We solved open loop simulations for each model with varying prediction horizons from 6 to 16. For better numerical tightness of results and avoiding of statistical errors, we did 10 repetitions of solution for each model on particular prediction horizon.

Results of the test are presented on figures 4.6 and 4.7, which are demonstrating considerable efficiency improvement of our YALMIP-FSM models with comparison to HYSDEL models, what is demonstrated on table 4.1. This improvement occurred because of YALMIP enhanced modeling techniques, where YALMIP during setting of constants (3.6) for Big-M models is considering parameters appearing in whole model, while HYSDEL during setting of Big-M constants (3.6) for resulting MIP problem is looking only at parameters defined in INTERFACE section of model code. Because of this, YALMIP models are numerically much tighter and therefore more efficient for solvers based on MIP relaxations, than models created via HYSDEL.

YALMIP-FSM vs HYSDEL	
Unary approach	~ 59 %
Binary approach	~ 56 %

Table 4.1: Improvement of efficiency of tested YALMIP-FSM models comparing to HYSDEL models.

Moreover from results of the test is obvious that models using unary encoding approach are providing better results than models with binary encoding approach. Demonstration of higher efficiency of unary encoding approach comparing to binary encoding approach is shown in following table 4.2. However these results are in conflict with primary assumptions about more efficient models with less binary variables by using binary encoding approach. Explanation comes with fact, that even if we were able to decrease the number of primary state binary variables logarithmically, the number of auxiliary binaries has raised in need to describe more complex logical expressions, what was mentioned in section 3.4 about Big-M modeling.

Unary Encoding vs Binary Encoding	
YALMIP-FSM	~ 27 %
HYSDEL	~ 33 %

Table 4.2: Comparison of tested models with unary encoding approach with models using binary encoding approach created in YALMIP-FSM and HYSDEL.

It is important to note, that differences in solver times for models are growing with rising prediction horizon, and therefore numbers presented in tables 4.1 and 4.2 are only estimations based on obtained data from this particular test.

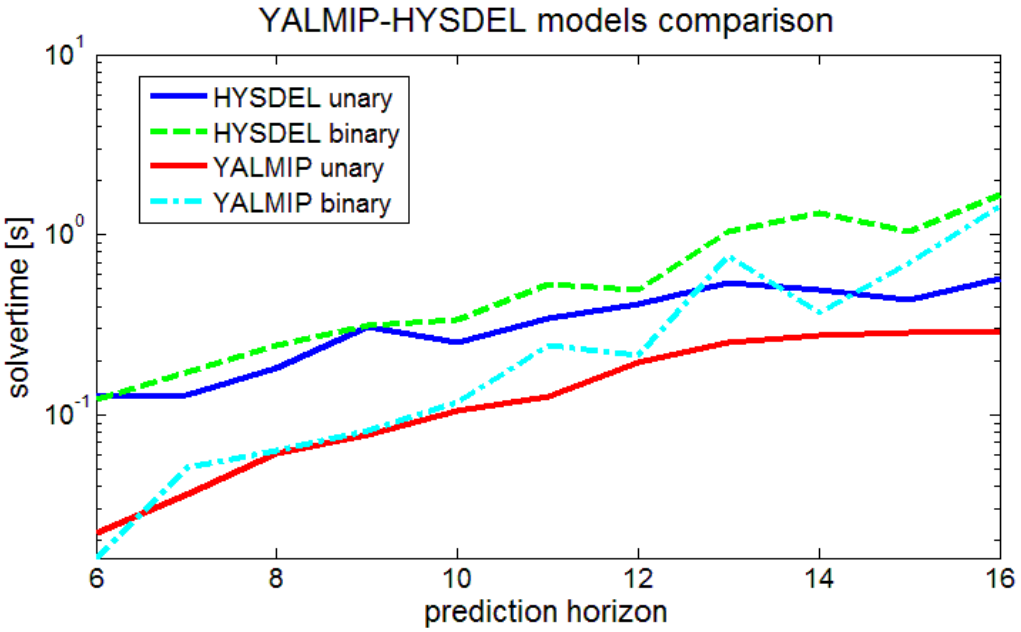

Figure 4.6: Comparison of solver times dependence on prediction horizon, for HYSDEL and YALMIP-FSM models with unary and binary encoding approaches.

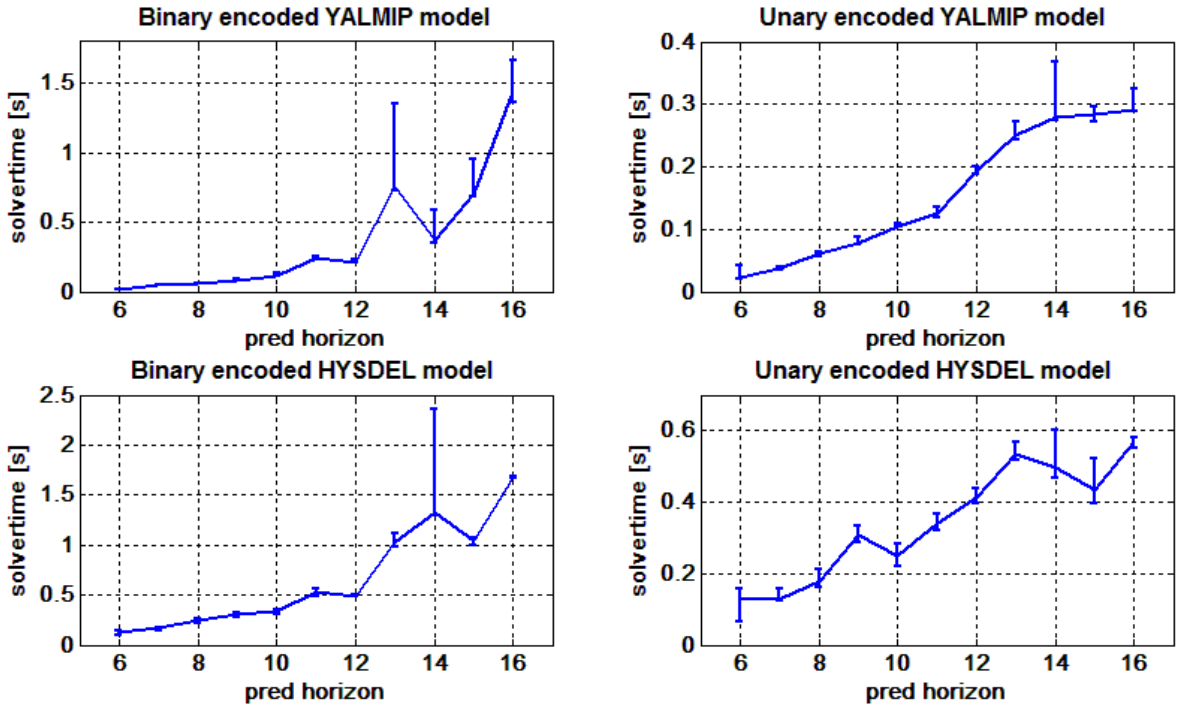

Figure 4.7: Maximal, minimal and average values of solver times for all tested models.

On figure 4.6 there visible that differences in solver times for particular models have approximately constant values, what means that complexity of models depend primary on prediction horizon. Figure 4.7 shows errorbars for tested models with maximal, minimal and average value of solver time for each prediction horizon, where we can observe few considerable deviations from average values, especially in using of binary encoding. General deviations were probably caused by random numerical drawbacks resulting from higher number of optimization problems solved repeatedly, while particularly high deviations for models with using binary encoding approach could be caused by more complex model construction, which is more sensitive on numerical tightness of corresponding MIP relaxations.

Our final recommendations based on test results are to keep using “basic” binary encoding approach (section 3.4), because these models proved to be more efficient than models using “enhanced” binary encoding approach (section 3.5), despite the initial assumptions.

Conclusion

As is well known hybrid models can be represented in form MIP problems, which belong to class of *NP*-hard problems. What roughly speaking means that there is no algorithm for solving this kind of problems in polynomial time, unless *P = NP*. Therefore efficiency improvement appears to be a crucial task in modeling of such systems, to provide solvable computational models.

In first part of the thesis we discussed DHA and PWA systems as theoretical modeling classes of hybrid systems, together with recently developed MLD systems as efficient highly computational oriented modeling framework for hybrid systems.

In second part of this thesis we are dealing with main goal of this thesis, which was to propose an efficient mathematical framework for modeling of hybrid systems represented either as FSM or PWA systems. For this purpose we introduced and discussed technique of Big-M modeling, which has proven to be efficient modeling technique for translation of hybrid models involving propositional logic statements into the MIP form. For enhanced modeling we proposed technique of "binary encoding" of state binary variables, which logarithmically decreases number of needed binary variables. Moreover we also introduced techniques for reducing the number of cuts (MIP inequalities) required for restricting infeasible combinations of auxiliary binaries from feasible region of resulting MIP model.

Third part of this thesis is introducing new hybrid modeling framework (YALMIP-FSM) based on modeling techniques mentioned above. This framework was developed as extension of free MATLAB optimization toolbox YALMIP, therefore is fully consistent with YALMIP and MATLAB syntax. We are receiving computational MIP model as an output of our framework, while hybrid system on input can be defined as DHA or PWA system. Moreover all low level modeling is done automatically and internally by YALMIP functions what gives user possibility to focus at high level modeling of hybrid systems. All these properties of YALMIP-FSM framework makes it extremely user friendly without additional need for knowing extra modeling languages or modeling features. Framework syntax was also demonstrated on two modeling examples.

In the end this thesis we computationally tested models created in YALMIP-FSM framework in comparison with models created in modeling language and compiler HYSDEL. We also tested enhanced modeling approach of binary encoding of state variables together with basic unary encoding approach. The results are demonstrating considerable efficiency improvement of our YALMIP-FSM models comparing to HYSDEL models. We obtained this improvement due to YALMIP enhanced modeling techniques which is constructing much tighter models, which are more suitable for solvers based on MIP relaxations. In our computational study we also obtain one surprising result, enhanced binary encoding approach has showed to be nearly one-third less efficient than basic unary encoding approach. This was probably caused by extra auxiliary binaries introduced internally into the model, necessary for description of more complex logical statements, which appeared with binary encoding approach.

Resumé

Hybridné systémy (HS) sú systémy obsahujúce súčasne spojité aj diskrétne dynamické správanie. V úvode tejto práce sú stručné predstavené základné vlastnosti takýchto systémov, pokus o ich rozdelenie do špecifickejších podskupín, spolu s príkladmi systémov z inžinierskej praxe. Vymenovávame najpoužívanejšie prístupy k modelovaniu hybridných systémov, z ktorých vyberáme a v druhej kapitole podrobnejšie opisujeme konkrétne diskrétne hybridné automaty (DHA) , po častiach afinné (PWA) systémy a výpočtovo orientované zmiešané logicko-dynamické (MLD) systémy.

V tretej kapitole detailne predstavujeme prístup k modelovaniu hybridných systémov a ich reprezentáciu vo forme celočíselného optimalizačného problému, ktorý používame v praktickej časti tejto práce. Uvádzame tri transformačné techniky na transformáciu logických výrazov do matematickej reprezentácie zmiešaného celočíselného optimalizačného problému, definovaného vo forme celočíselných lineárnych rovníc a nerovníc. Ďalej predstavujeme techniku takzvaného Big-M modelovania, slúžiacu na konštrukciu efektívnych modelov vo forme zmiešaného celočíselného programovania ako aj ich vzťah s modelovými prístupmi (DHA, PWA) spomenutými v druhej kapitole. Na tieto účely je predstavená knižnica Big-M modelov, pomocou ktorej sme schopný transformácie širokého spektra logických vzťahov a výrazov do príslušnej formy celočíselného programovania. Ako už bolo spomenuté, v probléme zmiešaného celočíselného programovania sú celočíselné premenné klasicky definované pomocou takzvaného unárneho (jednozložkového) kódovania, priraďujúc práve jednu binárnu premennú ku každej možnej celočíselnej hodnote. V poslednej sekcii tretej kapitoly sa venujeme efektívnejším prístupom k modelovaniu hybridných modelov, pomocou takzvaného binárneho kódovania, ktoré vyžaduje len logaritmické množstvo pôvodných binárnych premenných $log_2(N)$. Taktiež riešime problémy indikácie a separácie neriešiteľných kombinácií binárnych premenných pomocou extra ohraničení, takzvaných rezov pridávaných do modelu. Základný prístup spočíva podobne ako v unárnom kódovaní v priradení jedného ohraničenia (rezu) na separáciu práve jednej neriešiteľnej kombinácie. Pre zvýšenie efektivity výsledného modelu je však výhodné použitie menšieho počtu komplexných rezov, každý rez pre viacero neriešiteľných kombinácií súčasne. Preto predstavujeme techniku „redukovaných hlbokých rezov“ (reduced deep cuts approach) pomocou ktorej sme schopný minimalizovať

počet potrebných ohraničení približne na logaritmickú úroveň z počtu pôvodných ohraničení, pričom sa však súčasne snažíme vytvárať efektívne modely pre lineárne relaxácie celočíselného problému s čo možno najmenším počtom možných neceločíselných vrcholov v množine možných riešení pre daný model.

Štvrtá kapitola sa zaoberá programovou realizáciou modelových techník a postupov prezentovaných v kapitole tretej. Prezentujeme vytvorený programový balík (YALMIP-FSM) pre modelovanie hybridných systémov, ktorý je rozšírením voľne šíriteľného optimalizačného toolboxu YALMIP pre MATLAB. Je predstavená jednoduchosť novovytvoreného modelovacieho programového balíka, ktorého syntax je založená na syntaxe modelovacieho jazyka YALMIP, ktorý je konzistentný s jazykom MATLAB, tieto vlastnosti robia hybridné modelovanie pomocou YALMIP-FSM extrémne jednoduchým a intuitívnym pre bežného MATLAB používateľa.

Na záver demonštrujeme funkčnosť modelovacieho toolboxu YALMIP-FSM na dvoch príkladoch hybridných systémov. Taktiež je demonštrované porovnanie výsledných YALMIP-FSM modelov s modelmi vytvorenými pomocou modelovacieho jazyka HYSDEL. Výsledkom tohto porovnania, je fakt že modely vytvorené pomocou nami vytvoreného modelovacieho programového balíka YALMIP-FSM boli zhruba o 55% efektívnejšie ako HYSDEL (MLD) modely, čo sme dosiahli len vďaka vysoko efektívnym pokročilým modelovacím technikám a trikom obsiahnutých v toolboxe YALMIP. Avšak porovnanie rôznych prístupov kódovania binárnych premenných nám prinieslo mierne prekvapivé výsledky, keď sa zhruba o 30% ukázal byť efektívnejší základný prístup „unárneho“ kódovania oproti pokročilému prístupu „binárneho“ kódovania. Tento fakt si vysvetľujeme automatickým vnútorným priradením nových pomocných binárnych premenných toolboxom do modelu, ktoré sú potrebné na opis zložitejších logických výrazov, ktoré sa vyskytujú pri použití „binárneho“ kódovania.

Appendix

Appendix A: Figures and Codes

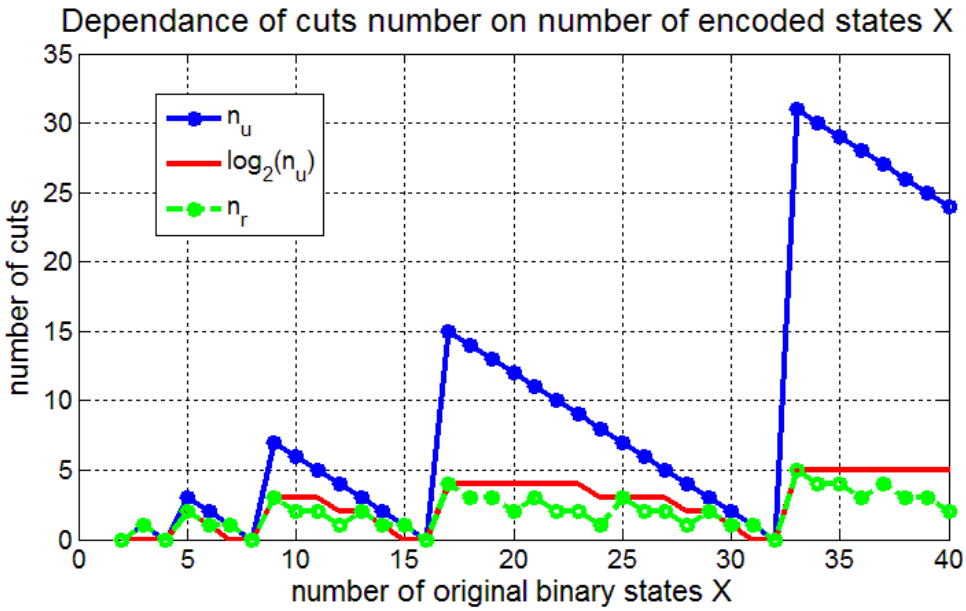

Figure A.1: Dependence of number of basic cuts n_u and number of reduced cuts n_r on number of original binaries X.

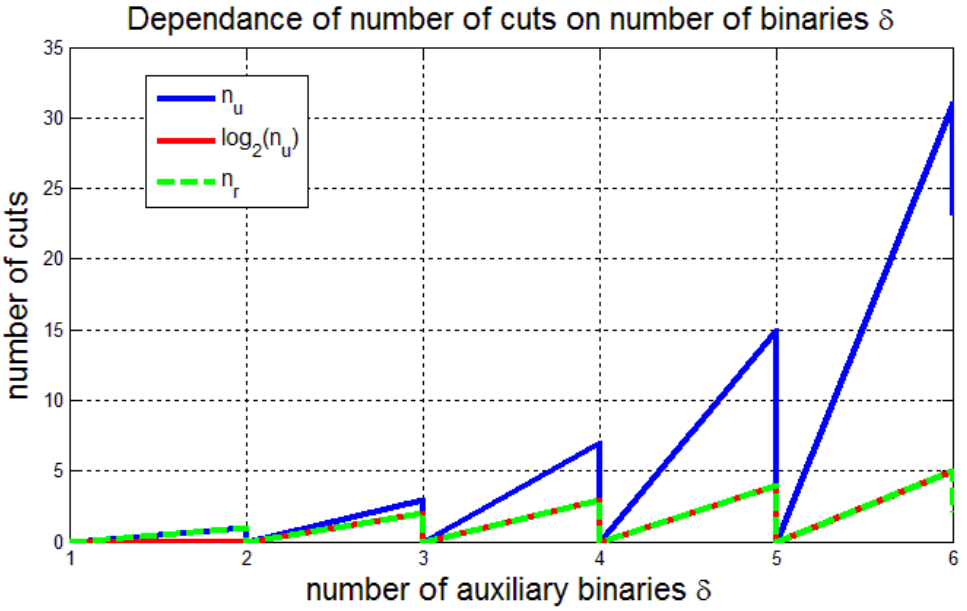

Figure A.2: Dependence of maximum number of basic cuts n_u and maximum number of reduced cuts n_r on number of auxiliary binaries delta.

Code A.1: YALMIP-FSM code for a truck delivery model from example 4.2:

```
% MODELING OF A TRUCK DELIVERY MODEL
N = 15;  % prediction horizon

% Declaration of variables
% x = (1)distance, (2)capacity
% u = (1)load, (2)unload, (3)speed
% mo = move
names = {'x' 'u' 'mo'};
varsizes = {[2,1] [3,1] [1,1] };
indicators = { 'r' 'r' 'b'};
%  params and bounds for variables
param.ineq = { ' [ -0.1*max_speed <= x{k}(1) <= (target_dis +
0.1*max_speed) , 0 <= x{k}(2) <= max_cap]';
'[0 <= u{k}(1)<=max_cap, 0 <= u{k}(2)<=max_cap , u{k}(3)>=0,
u{k}(3)<=max_speed]' };
param.eq = {};
param.val = { ' max_cap = 100; max_speed = 200; target_dis = 300' };
variables = yalmip_fsm_variables(names, varsizes, indicators, param );

% Declaration of states
s1 = yalmip_fsm_state('[x{k+1}(2) == x{k}(2)+u{k}(1), u{k}(2) <= 0,
u{k}(3) <= 0, x{k+1}(1) == x{k}(1) ]');
s2 = yalmip_fsm_state('[x{k+1}(1) == x{k}(1)+u{k}(3), u{k}(2) <= 0,
u{k}(1) <= 0, x{k+1}(2) == x{k}(2) ]');
s3 = yalmip_fsm_state('[x{k+1}(2) == x{k}(2)-u{k}(2), u{k}(1) <= 0,
u{k}(3) <= 0, x{k+1}(1) == x{k}(1) ]');
s4 = yalmip_fsm_state('[x{k+1}(1) == x{k}(1)-u{k}(3), u{k}(2) <= 0,
u{k}(1) <= 0, x{k+1}(2) == x{k}(2) ]');
states = [ s1, s2 , s3, s4];

% Declaration of transitions
t12 = yalmip_fsm_transition(s1, s2, '[ mo{k}(1) > 0.5 ]');
t23 = yalmip_fsm_transition(s2, s3, '[ x{k}(1) >= 300 ]');
t34 = yalmip_fsm_transition(s3, s4, '[ mo{k}(1) > 0.5 ]');
t41 = yalmip_fsm_transition(s4, s1, '[ x{k}(1) <= 0 ]');
trans = [t12, t23, t34, t41];

%----------- MODELING FUNCTION ---------------------------------
% 'unary' - basic approach , 'binary' - binary encoded states
Options.encoding = 'unary';
% cuts: 'basic' - one node one cut, 'enhanced' = reduced cuts
Options.cuts = 'basic';
% 0 - silent, 1 - elapsed time
Options.verbose = 1;
% core modeling function
[fsm, V] = yalmip_fsm(states, trans , variables, N, Options );
```

Code A.2: HYSDEL code for a unary encoded truck delivery model from example 4.2:

```
SYSTEM truck {
  /* basic model of a truck */
  INTERFACE {
    PARAMETER {
      REAL sampling = 1;
      REAL max_capacity = 1e2;
      REAL max_speed = 2e2;
      REAL eff_in = 1;
      REAL eff_out = 1;
      REAL target_distance = 300;
      }
    INPUT {
      REAL load [0, max_capacity];
      REAL unload [0, max_capacity];
      REAL speed  [0, max_speed];
      BOOL move ;
      }
    STATE {
      /* capacity off truck */
      REAL C [0, max_capacity];
      /* distance */
      REAL D [-0.1*max_speed, target_distance+0.1*max_speed];
      /*  loading = S1, unloading = S3, traveling = S2, returning = S4  */
      BOOL S1,S2,S3,S4;
      }
    }
  IMPLEMENTATION {
        AUX {
        /* capacities in individual states */
        REAL C1,C2,C3,C4;
        /* distances in individual states */
        REAL D1,D2,D3,D4;
        /* binary indicators for switching the states */
        BOOL d1,d2;
        /* binary variables for restrictions */
        BOOL no_load, no_unload, no_speed;
        /* auxiliary binary variables for state actualization */
        BOOL a1, a2, a3, a4;
        }
        AD {
             d1 = D <= 0;
             d2 = D >= target_distance;
             no_unload = unload <= 0;
             no_load = load <= 0;
             no_speed = speed <= 0;
        }
        LOGIC {
        a1 = (S1 & (~move)) | (S4 & d1);
        a2 = (S2 & (~d2)) | ( S1 & move ) ;
        a3 = (S3 & (~move)) | (S2 & d2) ;
        a4 = (S4 & (~d1)) | ( S3 & move ) ;
        }
        AUTOMATA {
        /*loading = S1, unloading = S3, traveling = S2, returning = S4 */
           S1 = a1;
           S2 = a2;
           S3 = a3;
           S4 = a4;
```

```
        }
        DA {
           C1 = { IF S1 THEN C+eff_in*load ELSE 0 } ;
           C2 = { IF S2 THEN C ELSE 0 } ;
           C3 = { IF S3 THEN eff_out*(C-unload) ELSE 0 } ;
           C4 = { IF S4 THEN C ELSE 0 }  ;
           D1 = { IF S1 THEN D ELSE 0 }  ;
           D2 = { IF S2 THEN D+sampling*speed ELSE 0 }  ;
           D3 = { IF S3 THEN D ELSE 0 } ;
           D4 = { IF S4 THEN D-sampling*speed ELSE 0 }  ;
        }
       CONTINUOUS {
         /* difference equations */
            C = C1 + C2 + C3 + C4;
            D = D1 + D2 + D3 + D4;
       }
       MUST {
        1 >= (REAL S1) + (REAL S2) + (REAL S3) + (REAL S4);
        1 <= (REAL S1) + (REAL S2) + (REAL S3) + (REAL S4);
        S1 -> no_unload;
        S2 -> no_unload;
        S4 -> no_unload;
        S2 -> no_load;
        S3 -> no_load;
        S4 -> no_load;
        S1 -> no_speed;
        S3 -> no_speed;
     }
  }
}
```

Code A.3: HYSDEL code for a binary encoded truck delivery model from example 4.2:

```
SYSTEM truck_binary {
  /* model of a truck with binary encoded states */
  INTERFACE {
    PARAMETER {
      REAL sampling = 1;
      REAL max_capacity = 1e2;
      REAL max_speed = 2e2;
      REAL eff_in = 1;
      REAL eff_out = 1;
      REAL target_distance = 300;
      }
    INPUT {
      REAL load [0, max_capacity];
      REAL unload [0, max_capacity];
      REAL speed  [0, max_speed];
      BOOL move ;
      }
    STATE {
      /* capacity off truck */
      REAL C [0, max_capacity];
        /* distance */
      REAL D [-2*max_speed, target_distance+2*max_speed];
      /* loading = S1, unloading = S3, traveling = S2, returning = S4*/
      /* variables for binary encoding of states: dA,dB
      S1 = ~dA&~dB
```

```
      S2 = ~dA&dB
      S3 = dA&~dB
      S4 = dA&dB  */
      BOOL dA,dB;
      }
    }
  IMPLEMENTATION {
    AUX {
    /* capacities in individual states */
    REAL C1,C2,C3;
    /* distances in individual states */
    REAL D1,D2,D3;
    /* conditions for switching the states */
    BOOL d1,d2;
    BOOL no_load, no_unload, no_speed;
    BOOL a1, a2;
    }
    AD {
         d1 = D <= 0;
         d2 = D >= target_distance;
         no_unload = unload <= 0;
         no_load = load <= 0;
         no_speed = speed <= 0;
    }
    LOGIC {
    a1 = ((~dA & dB) & d2) | (dA & ~dB) | ((dA & dB) & ~d1)  ;
    a2 =  ((~dA & ~dB) & move) | ((~dA & dB) & ~d2) | ((dA & ~dB) & move)
| ((dA & dB) & ~d1);
    }
    AUTOMATA {
           dA = a1;
           dB = a2;
    }
    DA {
       C1 = { IF (~dA & ~dB) THEN C+eff_in*load ELSE 0 } ;
       C2 = { IF  dB  THEN C ELSE 0 } ;
       C3 = { IF (dA & ~dB) THEN C-eff_out*unload ELSE 0 } ;
       D1 = { IF ~dB THEN D ELSE 0 }  ;
       D2 = { IF (~dA & dB) THEN D+sampling*speed ELSE 0 }  ;
       D3 = { IF (dA & dB) THEN D-sampling*speed ELSE 0 }  ;
    }
       CONTINUOUS {
      /* dif. eq of truck */
            C = C1 + C2 + C3 ;
            D = D1 + D2 + D3;
       }
     MUST {
     1 >= (REAL(~dA&~dB))+(REAL(~dA&dB))+(REAL(dA&~dB))+(REAL(dA&dB));
     1 <= (REAL(~dA&~dB))+(REAL(~dA&dB))+(REAL(dA&~dB))+(REAL(dA&dB));
        (~dA & ~dB) -> no_unload;
        (~dA & dB) -> no_unload;
        (dA & dB)  -> no_unload;
        (~dA & dB) -> no_load;
        (dA & ~dB) -> no_load;
        (dA & dB) -> no_load;
        (~dA & ~dB) -> no_speed;
        (dA & ~dB)  -> no_speed;
     }
  }
}
```

Appendix B: List of software on CD

YALMIP-FSM toolbox:

Core function: yalmip_fsm.m

States creation function: yalmip_fsm_state.m

Transitions creation function: yalmip_fsm_transition.m

Variables declaration function: yalmip_fsm_variables.m

Binary encoding function: binary_encoding_creation.m

Modeling code for a YALMIP-FSM thermostat model from example 4.1:

test_basic1.m

Modeling code for a YALMIP-FSM truck delivery model from example 4.2:

test_truck.m

Modeling code for a HYSDEL truck delivery models from example 4.2:

HYSDEL unary encoded model: truck.hys

HYSDEL binary encoded model: truck_binary.hys

References

[Agar] Ashish Agarwal - *Logical Modeling Frameworks for the Optimization of Discrete Continuous Systems*, Pittsburgh, PA, U.S.A.

[AJSMS01] A.J. van der Schaft and J.M. Schumacher. - *An Introduction to Hybrid Dynamical Systems*, volume 251 of Lecture Notes in Control and Information Sciences. Springer, London, 2000.

[AJSMS02] A.J. van der Schaft and J.M. Schumacher. - *Complementarity modelling of hybrid systems*. IEEE Transactions on Automatic Control, 1998.

[Al01] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho. - *Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems*. Springer Verlag, 1993.

[AlDi01] Alur, R. and Dill, D. L. - *A theory of timed automata, Theoretical Computer Science*, 1994.

[AnNe] Antsaklis, P. and Nerode, A. (eds): *IEEE Transactions on Automatic Control {*

Special Issue on Hybrid Control Systems, IEEE, 1998.

[Ant01] P.J. Antsaklis. - *A brief introduction to the theory and applications of hybrid systems.*

Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications, July 2000.

[ASan01] Adrian Sandu - *Computer Representation of Numbers and Computer Arithmetic*, February 2008.

[Asar01] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. - *Data-structures for the verification of timed automata*. In O. Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*. Springer Verlag, 1997.

[BaGu01] Back, A., Guckenheimer, J., and Myers, M. - *A dynamical simulation facility for hybrid systems.* In: Grossman, R., Nerode, A., Ravn, A., and Rischel, H. (eds) Hybrid Systems. Springer, New York, 1993.

[BarPan01] Barton, P. I. and Pantelides, C. C. - *Modeling of combined discrete-continuous processes*, AIChE Journal, 1994.

[BemDHS] A.Bemporad – *Modeling, Control, and Reachability Analysis of Discrete-Time Hybrid Systems*, March 25, 2003

[BemMor] A.Bemporad , Morari – *Control systems integrated logic, dynamics and constraints*, Automatica, 1999.

[BemMPC] A.Bemporad – *Model Predictive Control of Hybrid Systems Lectures*

[BeToMo] A. Bemporad, F.D. Torrisi, and M. Morari. - *Discrete-time hybrid modeling and verification of the batch evaporator process benchmark*. European Journal of Control, July 2001.

[BoyVan] Stephen Boyd , Lieven Vandenberghe - *Convex Optimization*, Cambridge University Press 2004.

[Bran] Michael S. Branicky - *Introduction to Hybrid Systems*, Cleveland, OH 44106, U.S.A.

[Bro01] Brockett, R.W. - *Hybrid models for motion control systems*. In: Trentelman, H.L., and Willems, J.C. (eds) Essays in Control, Boston, 1993.

[CaHeS] M.K. Camlıbel, W.P.M.H. Heemels, and J.M. Schumacher. - *Consistency of a time-stepping method for a class of piecewise linear networks*. To appear in *IEEE Trans. Circuits Syst. – I*, 2002.

[CaPaSo] Cavalier, T., Pardalos, P. and Soyster, A. - *Modeling and integer programming techniques applied to propositional calculus*, Computers and Operations Research, 1990.

[Eqvival] *Equivalence of hybrid dynamical Models* - W.P.M.H. Heemels, B. De Schutter, and A. Bemporad, Automatica, vol. 37, no. 7, July 2001.

[FTMLM03] Ferrari-Trecate, G., M. Muselli, D. Liberati and M. Morari - *A clustering technique for the identification*

[Fuku01] K. Fukuda. - *cdd/cdd+ Reference Manual*. Institute for Operations Research, ETHZentrum, CH-8092 Zurich, Switzerland, December 1997.

[Hay01] Hayes, J. - *Introduction to Digital Logic Design*, Addison-Wesley Publishing Company, Inc. 1993.

[He01] W.P.M.H. Heemels. - *Linear complementarity systems: a study in hybrid dynamics*. PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, 1999.

[HeSB01] W.P.H.M Heemels, B. de Schutter, and A. Bemporad. - *On the equivalence of classes of hybrid dynamical models*. In Proc. 40th IEEE Conf. on Decision and Control, Orlando, Florida, 2001.

[HeSchW] W.P.M.H. Heemels, J.M. Schumacher, and S.Weiland. - *Linear complementarity systems.* SIAM Journal on Applied Mathematics, 2000.

[Hyb] http://control.ee.ethz.ch/~hybrid/

[Hys] *HYSDEL 2.0.6 – User Manual* - Fabio Danilo Torrisi, Alberto Bemporad, Gioele Bertini, Peter Hertach, Dominic Jost, Domenico Mignone October 10, 2011

[ChHoo01] V. Chandru and J.N. Hooker. - *Optimization methods for logical inference*. Wiley-Interscience, 1999.

[Chr01] D. Christiansen. - *Electronics Engineers' Handbook, 4th edition*. IEEE Press/ Mc- Graw Hill, Inc., 1997.

[Chris] Frank J. Christophersen - *Optimal Control and Analysis for Constrained Piecewise Affine Systems*, ETH Zurich, 2006.

[KGBC] Kvasnica, M., P. Grieder, M. Baotic and F. J. Christophersen - *Multi-Parametric Toolbox (MPT)*. Automatic Control Laboratory, ETH Zurich, March 2006.

[Koh] Kohavi, Z. - *Switching and Finite Automata Theory*, McGraw-Hill, 1978.

[Lys] John Lygeros - *Lecture Notes on Hybrid Systems*, Department of Electrical and Computer Engineering, University of Patras, Greece, 2004

[LysTom] John Lygeros, Shankar Sastry, and Claire Tomlin – *Hybrid Systems: Foundations, advanced topics and applications*, April 29, 2010

[Med01] Mendelson, E. - *Introduction to mathematical logic*, Van Nostrand, 1964.

[Mig] Domenico Mignone – *Control and estimation of hybrid systems*, ETH Zurich, January, 2002.

[MoBeMi02] M. Morari, A. Bemporad, and D. Mignone - *A framework for control, state estimation, fault detection, and verification of hybrid systems*, 1999.

[Olaru] S. Olaru, I. Prodan, F. Stoican - *Enhancements on the hyperplane arrangements in mixed integer techniques***,** 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC) Orlando, FL, USA, December 12-15, 2011

[Parbar01] T. Park and P.I. Barton - *Implicit model checking of logic-based control systems*. AIChE Journal, 1997.

[PoMuZu] B. Potocnik, G. Music, B. Zupancic - *A New Technique for Translating Discrete Hybrid Automata into Piecewise Affine Systems*, Mathematical and Computer Modelling of Dynamical Systems 2004.

[RBL04] Roll, J., A. Bemporad, L. Ljung: *Identification of piecewise affine systems*

via mixed-integer programming. Automatica, 2004.

[Sontag] E.D. Sontag - *Nonlinear regulation The piecewise linear approach*. IEEE Transactions on Automatic Control, 26(2):346–358, April 1981.

[SchM02] B. De Schutter and B. De Moor - *The extended linear complementarity problem and the modeling and analysis of hybrid systems.* In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*. Springer, 1999.

[SchB01] B. De Schutter and T. van den Boom - *On model predictive control for max-min-plus-scaling discrete event systems*. Automatica, , 2001.

[Silv01] B.I. Silva, O. Stursberg, B.H. Krogh, and S. Engell - *An assessment of the current status of algorithmic approaches to the verification of hybrid systems.* In Proc. 40th IEEE Conf. on Decision and Contro*l*, Orlando, Florida, December 2001.

[Sch01] B. De Schutter - *Optimal control of a class of linear hybrid systems with saturation*. SIAM J. Control Optim., 2000.

[ToBe] Fabio Danilo Torrisi and Alberto Bemporad – *HYSDEL - A Tool for Generating Computational Hybrid Models for Analysis and Synthesis Problems,* IEEE Transactions on control systems technology, March 2004

[Torr] Fabio Danilo Torrisi - *Modeling and Reach-Set Computation for Analysis and Optimal Control of Discrete Hybrid Automata*, ETH Zurich, 2003.

[wikiWP01] http://wiki.math.toronto.edu/DispersiveWiki/index.php/Well-posedness

[wikiWP02] http://en.wikipedia.org/wiki/Well-posed_problem

[wikiNPh] http://en.wikipedia.org/wiki/NP-hard

[wikiCNF] http://en.wikipedia.org/wiki/Conjunctive_normal_form

[wikiCoNu] http://en.wikipedia.org/wiki/Computer_number_format

[wikiTT] http://en.wikipedia.org/wiki/Truth_table

[Wil01] Williams, H. - *Model Building in Mathematical Programming*, John Wiley & Sons, Third Edition, 1993.

[YAL] http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Main.HomePage

[YalBigM] http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.Big-M

[YalMIP] http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.MixedInteger Representations

[ZhBrPh] Zhang, W., Branicky, M.S., and Phillips, S.M. - *Stability of networked control systems.* IEEE Control Systems Magazine, 2001.