Linköping Studies in Science and Technology. Dissertations
No. 652

Model Checking Methods for Mode Switching Systems

Valur Einarsson

Department of Electrical Engineering
Linköpings universitet, SE–581 83 Linköping, Sweden

Linköping 2000

Model Checking Methods for Mode Switching Systems

`valure@isy.liu.se`
`http://www.control.isy.liu.se`
Department of Electrical Engineering
Linköpings universitet
SE–581 83 Linköping
Sweden

ISBN 91-7219-836-2 ISSN 0345-7524

Printed by Linus & Linnea AB, Linköping, Sweden 2000

Abstract

This thesis deals with modeling and analysis of a special class of hybrid systems, i.e., systems displaying behavior of both continuous and discrete nature.

We focus on the class of systems which are naturally modeled as having continuous trajectories, but where the trajectories evolve according to a system of first order differential equations with piecewise continuous right hand side. We further restrict our attention to models where the right hand side is piecewise affine in the continuous state variables. Systems which are naturally modeled in this fashion are called piecewise affine switched systems.

The modeling is packaged into a modeling framework where the right hand side of the differential equations consists of both continuous and discrete valued variables. The equations then run in combination with a finite state machine, and their interaction is specified via an interface. This framework allows a compact description of a potentially large number of affine models, as well as a modular approach to modeling of complex systems.

The analysis consists of verifying if a switched system fulfills specifications given either in terms of sets of good or bad discrete states, or as formulas in temporal logic specifying desired or undesired behavior. This kind of analysis is often applied to purely discrete systems. We therefore take the approach to discretize the switched system using conservative approximations. This is done using two different automated procedures devised in this thesis. Both methods can be seen as special instances of using invariant sets and Lyapunov theory to guarantee either that bad states are avoided and/or that goal states are reached in finite time.

The former discretization method results in two different approximations, named acceptor and exceptor, which together can be utilized to obtain bounds on behavior which can be certified. We also introduce a discrete device, called reflector, which allows us to reduce the non-determinism resulting from the approximations. Furthermore, implementations of the procedure suggested, using either linear programming or quantifier elimination as computational tools, are discussed.

The second method utilizes invariant sets of a special type called power cones, of which quadratic sets such as cones and paraboloids are special instances. This approach results in tighter approximations while still allowing for efficient implementation using Linear Matrix Inequalities and convex optimization.

Together, the discrete approximations, obtained automatically from a model of a switched system, can be used to verify properties of interest. This is illustrated as we apply one of our methods to two examples, a chemical reactor model and a model of the landing gear of a fighter aircraft.

Acknowledgments

I am deeply grateful to all the people who have supported and encouraged me during the writing of this thesis.

First, I would like to thank Prof. Torkel Glad and Prof. Lennart Ljung for their excellent guidance during this work.

Several people have read versions of the manuscript at different stages of preparation and given valuable feedback. My thanks go to Dr. Krister Edström, Dr. Johan Gunnarsson, Lic. Eng. Mikael Norrlöf and Jacob Roll. Your suggestions and comments have certainly improved the quality of the thesis and are greatly appreciated. I would also like to thank Mikael, Jacob, Johan Löfberg and Fredrik Tjärnström for our discussions and for patiently answering my questions.

The Automatic Control group at Linköpings universitet is a collection of people which together create a unique atmosphere. Being in that atmosphere has been a great source of inspiration, as well as distraction through the fika-room discussions and floorball games.

Conducting research in the zone between systems engineering and computer science has been greatly assisted through the ECSEL graduate school which has provided valuable sources of knowledge and inspiriation which is hereby acknowledged.

This work was supported by the Swedish Research Council for Engineering Sciences (TFR), which is gratefully acknowledged.

Linköping, August 2000

Valur Einarsson

Contents

1 Introduction

Dynamic systems are a natural part of our surroundings although we do not always recognize them as such. However, in an engineering context, they become more apparent and we are thus motivated to study them.

Traditionally, the treatment of dynamic systems has taken place mainly in the control and systems theory community as well in some disciplines of mathematics. More recently, computer scientists have approached dynamic systems, extending the arsenal available for analysis and design with additional tools.

It is in this zone of overlap between systems theory and computer science that the concept of hybrid systems appears. Hybrid systems are systems which are naturally modeled as displaying a combination of continuous and discrete behavior.

The topic has been an active field of research in recent years. However, and in spite of the fact that many engineering systems belong to this class, systematic methods for analysis and control of relatively general hybrid systems are still to appear.

The main objective with this thesis is presenting a methodology for modeling and algorithmic analysis of a class of hybrid systems.

We focus on the class of systems which are naturally modeled as having continuous trajectories, but where the trajectories evolve according to a system of first order differential equations with piecewise continuous right hand side. We further restrict our attention to models where the right hand side is piecewise affine in the continuous state variables. Systems which are naturally modeled in this fashion are called piecewise affine switched systems.

The modeling is packaged into a modeling framework where we take a logical approach to writing down the right hand side of the differential equations. The equations then run in combination with a finite state machine, and their interaction is specified via an interface. This framework allows a compact description of a potentially large number of affine models, as well as a modular approach to modeling of complex systems.

The analysis consists of verifying if a switched system fulfills specifications given either in terms of sets of good or bad discrete states, or as formulas in temporal logic specifying desired or undesired behavior. The verification is performed automatically, given a system model and a specification, this kind of analysis is commonly known as model checking and is often applied to purely discrete systems. We therefore take the approach to discretize the switched system using conservative approximations. This is done using two automated procedures devised in this thesis.

The former discretization results in two different approximations, named acceptor and exceptor, which together can be utilized to obtain bounds on behavior which can be certified. We also introduce a discrete device, called reflector, which allows us to reduce the non-determinism resulting from the approximations. Furthermore, implementations of the procedure suggested, using either linear programming or quantifier elimination as computational tools, are discussed.

The latter discretization utilizes a specific type of invariant sets having an indicator function which is quadratic in all variables except one where it is a power function. We refer to these sets as power cones and describe procedures for representing and operating on them as well as their use for approximation.

Together, the discrete approximations, obtained automatically from a model of a switched system, can be used to verify properties of interest. This is illustrated as we apply our methods to two examples, a chemical reactor model and a model of the landing gear of a fighter aircraft.

1.1 Thesis Outline

The layout of this thesis is as follows. In Chapter 2 we present some basic concepts from set theory which are utilized in the thesis. In Chapter 3, affine and quadratic sets are presented, we discuss how they are represented and operated on. Chapter 4 presents the sets we refer to as power cones and introduces methods for representation of the sets themselves as well as operations such as intersection and inclusion, also involving affine and quadratic sets. These are essential for the application of these sets in the thesis.

In Chapter 5 we discuss dynamic systems, continuous, discrete and hybrid, and their modeling, introducing a modeling framework for the class of piecewise linear switched systems.

In Chapter 6, we describe analysis methods for discrete event systems with the emphasis on model checking which we utilize as an analysis method for switched systems in this thesis. Chapter 7 contains some analysis methods for continuous systems, these are also an essential part of our treatment of systems having both discrete and continuous elements. In Chapter 8, we describe how model checking is performed for switched systems in an ideal situation and discuss the effect of approximations on that procedure. In Chapters 9 and 10 we describe the two methods for obtaining conservative abstractions of switched systems and how they are used to perform model checking of switched systems. These two methods are the main contribution of this thesis

Finally, in Chapter 11, we apply one of our methods to two examples in order to illustrate their use.

1.2 Contributions

The main contributions of this thesis are the following.

- The modeling formalism for piecewise affine switched systems, presented in Section 5.2.
- The introduction of the power cones of Chapter 4 and methods for approximatively representing their intersection with affine sets and determining whether they have polytopes and quadratic sets as subsets.
- The definition of transition conditions for a set and the use of reachable sets and regions of attraction and their conservative approximations for determining those conditions. These are found in the lemmas of Chapter 7.
- The methods for performing automatic discretization of a switched model using conservative approximations. These are described in Chapters 9 and 10 and are the basis for performing model checking for switched systems.
- The use of power cones as invariant sets for affine systems, in particular Lemma 10.2.
- Transforming logical formulas involving relations over real variables, as well as discrete variables, into a purely logical expression using quantifier elimination is a novel application of this computational tool. This is suggested in Section 9.6.
- The application of one of the model checking methods to examples of reasonably large size, compared to the ones treated in the hybrid systems literature so far.

Part I

Preliminaries

2

Sets and Relations

We will be dealing with both finite and infinite domains. The formalism of sets and relations provides a convenient way of handling both and we therefore give some formal definitions and describe their use in the respective domains.

We start with a summary of concepts from set theory, discuss how sets can be represented in different ways and discuss the use of coding to represent finite sets. The material in this chapter serves as a compact tutorial on subjects utilized in this thesis and contains no new results.

2.1 Concepts from Set Theory

Our use of set theoretical concepts is standard and can be found in textbooks such as [21].

A set $\mathcal{S}$ is a (finite or infinite) collection of objects. We can represent finite sets using either explicit listings or implicit definitions. An explicit listing has the form $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, where each $s_i \in \mathcal{S}$ is an element of $\mathcal{S}$. An implicit definition is of the form $\mathcal{S} = \{z \in \Omega \mid S(z)\}$ where Ω is the universal set and z is a variable having as values those elements from Ω that satisfy the constraint (indicator function) $S : \Omega \to \mathbb{B}$ where $\mathbb{B}$ is the set $\{\texttt{true}, \texttt{false}\}$. In the case of infinite sets, we of course need to rely on implicit representations. These also turn out to be very convenient when dealing with finite sets.

Note that the constraint $S(z)$ carries all relevant information about the set $\mathcal{S}$ for a given universal set. Hence, relationships between sets can be expressed in terms of conditions on the constraints.

Definition 2.1 (Relationships Between Sets)
For the sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \Omega$, we have the relationships shown in Table 2.1. In addition,

Relationship	Notation	Condition
Element	$s \in \mathcal{S}_1$	$S_1(s)$
Subset	$\mathcal{S}_1 \subseteq \mathcal{S}_2$	$S_1(z) \to S_2(z)$
Equal	$\mathcal{S}_1 = \mathcal{S}_2$	$S_1(z) \leftrightarrow S_2(z)$
Universal set	Ω	`true`
Empty set	$\emptyset$	`false`

Table 2.1: Set relationships

the set of all subsets of $\mathcal{S}_1$ is called the power set *of $\mathcal{S}_1$ and denoted $2^{\mathcal{S}_1}$.*

Furthermore, due to the isomorphism of set algebra and propositional logic, set operations can be represented by logic operations on the constraints.

Definition 2.2 (Set Operations)
For the sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \Omega$, we have the operations shown in Table 2.2.

Operation	Notation	Definition
Union	$\mathcal{S}_1 \cup \mathcal{S}_2$	$\{z \in \Omega \mid S_1(z) \vee S_2(z)\}$
Intersection	$\mathcal{S}_1 \cap \mathcal{S}_2$	$\{z \in \Omega \mid S_1(z) \wedge S_2(z)\}$
Set minus	$\mathcal{S}_1 \setminus \mathcal{S}_2$	$\{z \in \Omega \mid S_1(z) \wedge \neg S_2(z)\}$
Complement	$\overline{\mathcal{S}_1}$	$\{z \in \Omega \mid \neg S_1(z)\}$

Table 2.2: Set operations

Let us briefly exemplify the use of logical expressions to describe sets and set operations.

Example 2.1 (Finite sets) *For the natural numbers, $\Omega = \mathbb{N}$, we can define the sets*

$$\mathcal{S}_1 = \{3, 4, 5\} = \{z \in \mathbb{N} \mid 3 \le z \le 5\} \tag{2.1}$$

$$\mathcal{S}_2 = \{z \in \mathbb{N} \mid z \ge 2\} \tag{2.2}$$

The set $\mathcal{S}_1$, represented by the formula $S_1(z) = [3 \le z \le 5]$ is finite while the set represented by $S_2(z) = [z \ge 2]$ is infinite (but countable). We see that $S_1(z) \to$

$S_2(z)$ and hence $\mathcal{S}_1$ is a subset of $\mathcal{S}_2$. Furthermore, the set

$$\begin{aligned}\overline{\mathcal{S}}_1 &= \{z \in \mathbb{N} \mid \neg S_1(z)\}\\ &= \{z \in \mathbb{N} \mid \neg[3 \leq z \leq 5]\}\\ &= \{z \in \mathbb{N} \mid \neg\big[[z \geq 3] \wedge [z \leq 5]\big]\}\\ &= \{z \in \mathbb{N} \mid [z < 3] \vee [z > 5]\}\end{aligned} \tag{2.3}$$

intersected with $\mathcal{S}_2$ yields the set

$$\begin{aligned}\overline{\mathcal{S}}_1 \cap \mathcal{S}_2 &= \{z \in \mathbb{N} \mid \neg S_1(z) \wedge S_2(z)\}\\ &= \{z \in \mathbb{N} \mid \big[[z < 3] \vee [z > 5]\big] \wedge [z \geq 2]\}\\ &= \{z \in \mathbb{N} \mid [z < 3] \wedge [z \geq 2] \vee [z > 5] \wedge [z \geq 2]\}\\ &= \{z \in \mathbb{N} \mid [z = 2] \vee [z > 5]\}\end{aligned} \tag{2.4}$$

We now turn our attention to relations, but first we need an operator which generates a set of ordered tuples containing elements from given sets.

Definition 2.3 (Product)
For sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \Omega$ the Cartesian product, or cross product, of $\mathcal{S}_1$ and $\mathcal{S}_2$ is denoted by $\mathcal{S}_1 \times \mathcal{S}_2$ and defined by

$$\mathcal{S}_1 \times \mathcal{S}_2 = \{(z_1, z_2) \mid S_1(z_1) \wedge S_2(z_2)\} \tag{2.5}$$

For sets $\mathcal{S}_1, \ldots, \mathcal{S}_n \subseteq \Omega$ their product is denoted $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and equals

$$\{(z_1, \ldots, z_n) \mid \bigwedge_{1 \leq i \leq n} S_i(z_i)\} \tag{2.6}$$

We now define subsets of the product sets obtained.

Definition 2.4 (Relations)
A relation on $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ is any subset of the product set of $\mathcal{S}_i, \mathcal{I}$, i.e.

$$\mathcal{R} \subseteq \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \tag{2.7}$$

is a relation on the set $\mathcal{S}_i \times \cdots \times \mathcal{S}_n$. This subset can of course be represented implicitly as

$$\mathcal{R} = \{(z_1, \ldots, z_n) \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \mid R(z_1, \ldots, z_n)\} \tag{2.8}$$

where $R : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathbb{B}$ is a multivariable constraint representing the relation.

We will make frequent use of particular kinds of sets and relations throughout the thesis. They are introduced in the next chapter.

2.2 Representation of Sets

When dealing with sets in $\mathbb{R}^n$, we commonly use different representations for the same set. Our choice of representation usually reflects the intended manipulations and we normally have a way of transforming between representations. The sets we will be working with can generally be described by

$$\mathcal{S} = \{x = f(z) \mid g(z) \leq 0 \wedge h(z) = 0\} \tag{2.9}$$

where f is a function, $g(z) \leq 0$ denotes a number of inequalities and $h(z) = 0$ is a number of equalities. In [32] this description is called a constrained parameter representation.

There are two special cases where either f, or g and h, are missing (or trivial). A set of the form

$$\mathcal{S} = \{x = f(z) \mid z \in \mathbb{R}^m\} \tag{2.10}$$

is called a free parameter representation since there are no constraints on z. On the other hand, the description

$$\mathcal{S} = \{x \in \mathbb{R}^n \mid g(x) \wedge h(x)\} \tag{2.11}$$

is called a constraint representation since x is constrained by equations and inequalities.

Example 2.2 *Consider the translated cylinder in 3-dimensional space which can be represented using the constraint representation*

$$\mathcal{S} = \{x \in \mathbb{R}^3 \mid (x_1 - 1)^2 + (x_2 - 1)^2 \leq 1 \wedge 0 \leq x_3 \leq 1\} \tag{2.12}$$

We can also choose a a constrained parameter representation of the same set, resulting in

$$\mathcal{S} = \{x = f(z) \mid -\pi \leq z_1 \leq \pi \wedge 0 \leq z_2 \leq 1 \wedge 0 \leq z_3 \leq 1\} \tag{2.13}$$

where $f : \mathbb{R}^3 \to \mathbb{R}^3$, $f(z) = (z_2 \cos z_1, z_2 \sin z_1, z_3)^T$ *represents the transformation to cylindrical coordinates.*

Similarly, when dealing with finite sets and relations, it may be convenient to represent the sets in terms of transformation of variables taking values from different sets. We then speak of codes, that is, we represent the finite set $\mathcal{S}$ as

$$\mathcal{S} = \{s = \lambda(x) \mid x \in \Omega_x^n,\ S(x)\} \tag{2.14}$$

where the variables x are said to encode the set $\mathcal{S}$ and, conversely, the elements $s \in \mathcal{S}$ can be seen as the interpretation of a variable x.

Formally, we have the following definition [17].

Definition 2.5 (Code)
A code C is a triple $(\mathcal{X}, \mathcal{S}, \lambda)$ where $\mathcal{X} = \{x^1, \dots, x^r\}$ is a set of code words, where each element x_i of an n-dimensional vector x from $\mathcal{X}$ takes its value from the set Ω_x, i.e., $x \in \Omega_x^n$. Furthermore, $\mathcal{S} = \{s^1, \dots, s^r\}$ is a set of symbols, and $\lambda : \mathcal{X} \to \mathcal{S}$ is an injective code function defining the coding of symbols.

Remark 2.1
Note that when we add subscripts to vector x, say x_i, we are referring to the i-th element of the vector x. But when we add superscripts, for instance by writing x^j, we mean a specific vector labeled j. We may thus refer to a specific vector and its elements as $x^j = (x_1^j, x_2^j, \dots, x_r^j)^T$.

The explicit use of codes usually appears in the context of digital communications, see [26] and [5], but they are also frequently utilized for representing finite sets in other contexts and we will find them useful in our applications since they allow compact representations of large sets. Consider, for instance, the set $\mathcal{S} = \{s^0, s^1, \dots, s^5\}$. This set may be represented by a three-dimensional Boolean vector using an appropriate code function. This is illustrated in Figure 2.1. Now,

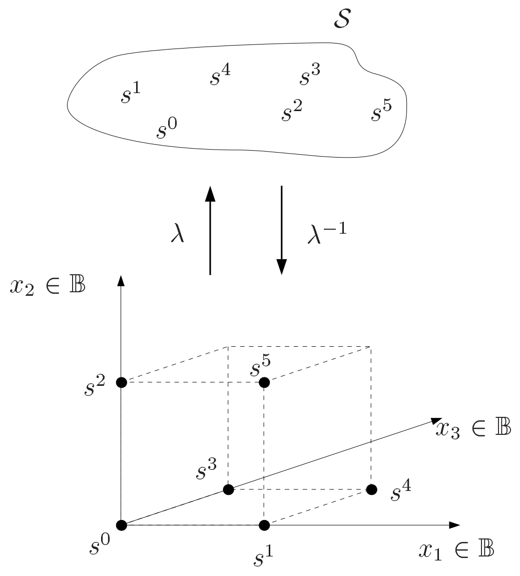


Figure 2.1: Reducing complexity using codes.

encoding six elements using three variables is no significant achievement. However, the situation is quite different if we allow code words of length, say 100, enabling us to represent implicitly up to 2^{100} states!

Let us take a closer look at how we may define a code.

Example 2.3 (A Code) *Consider the set $\mathcal{S} = \{red, yellow, green\}$. This set can be coded in the variables $x_1, x_2 \in \mathbb{B}$ using the function defined in Table 2.3 and visualized in Figure 2.2.*

$\lambda(x)$		x_2	
		`false`	`true`
x_1	`false`	–	*red*
	`true`	*green*	*yellow*

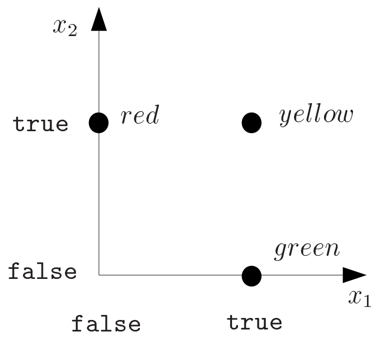


Figure 2.2: Visualization of a code.

We can thus write the set as $\mathcal{S} = \{\lambda(x) \mid x \in \mathbb{B}^2,\ x_1 \vee x_2\}$ and use the expression $S(x) = [x_1 \vee x_2]$ when performing set operations. The results can then be interpreted in terms of the original elements using the code function $\lambda(x)$. For instance, the result of excluding green from $\mathcal{S}$ can be obtained as

$$\begin{aligned} \mathcal{S} \setminus \{green\} &= \{s = \lambda(x) \mid x \in \mathbb{B}^2,\ [x_1 \vee x_2] \wedge \neg[x_1 \wedge \neg x_2]\} \\ &= \{s = \lambda(x) \mid x \in \mathbb{B}^2,\ [x_1 \vee x_2] \wedge [\neg x_1 \vee x_2]\} \qquad (2.15) \\ &= \{s = \lambda(x) \mid x \in \mathbb{B}^2,\ x_2\} = \{red, yellow\} \end{aligned}$$

When the code is defined over an alphabet consisting of the first q natural numbers, i.e., $\Omega_x = \{0, 1, \ldots, q-1\}$, C is called a q-ary code [5]. It is then common to define a (Hamming) distance between two code words, say x and y, as the number of different elements in x and y [5]. We will find it convenient to use codes over alphabets containing also negative integers. Furthermore, we need a non-standard definition of the distance between two code words.

Definition 2.6 (Distance)
The distance between two code words, $x = (x_1, \dots, x_n)^T$ and $y = (y_1, \dots, y_n)^T$, where $x, y \in \mathbb{Z}^n$, is defined as

$$d_c(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.16}$$

where $|\cdot|$ denotes the absolute value.

The distance thus measures the sum of absolute differences between values of each position. This is in contrast with the usual Hamming distance between code words which only measures the number of positions which differ. Obviously, the measures coincide in the case of binary codes.

Example 2.4 (Distance) *The distance between the binary code words $x = (0, 1, 0)^T$ and $y = (1, 1, 1)^T$ is $d_c(x, y) = 1 + 0 + 1 = 2$. The distance between the code words $x = (-1, 1, 0, 1)^T$ and $y = (0, 1, -1, -1)^T$ is $d_c(x, y) = 1+0+1+2 = 4$.*

In the following chapters we will see several specific examples of sets and relations and look at how they can be represented and operated on in an efficient manner. This allows us to use these concepts for developing automated computational methods for verification.

3 Affine and Quadratic Sets

The model checking methods for switched systems presented in Chapters 9 and 10 rely on the use of a certain type of continuous sets described in Chapter 4. Representation of and operations on these sets utilize approximations using sets defined by affine and quadratic equality and inequality constraints. We will need ways of representing these sets as well as methods for performing efficient operations such as feasibility and set inclusion. This chapter is devoted to defining these sets and presents procedures for operating on them. The material presented here, with the exception of Lemma 3.5, can be found in the thesis [32].

We start by giving definitions of some affine and quadratic sets and present how these can be represented. We then present some results on feasibility of such sets, i.e., how to determine whether such sets are empty or not. Finally we state some conditions that can be used to check if some specific sets are subsets of other sets.

3.1 Representation

In this section we introduce sets that we will make frequent use of in this thesis and discuss how they can be represented.

3.1.1 Affine Sets

Here we will consider relations over the reals which can be described using affine equations and inequalities. That is, the relations can be expressed using formulas $R : \mathbb{R}^n \to \mathbb{B}$ constructed, using logical connectives, from atomic formulas involving affine functions, $[r(x)\ \rho\ 0]$, with $r(x) = a_0 + \sum_{i=1}^{n} a_i x_i$ and $\rho \in \{=, \leq, \geq, <, >, \neq\}$.

The simplest constructions we will form using affine relations are hyperplanes and halfspaces. From a finite number of these we can then construct a bit more interesting geometric objects.

Definition 3.1 (Hyperplanes and Halfspaces)
A hyperplane in $\mathbb{R}^n$ is a set of the form

$$\mathcal{H} = \{x \in \mathbb{R}^n \mid c^T(x - x_0) = 0\} \tag{3.1}$$

where $c, x_0 \in \mathbb{R}^n$. Equivalently,

$$\mathcal{H} = \{x = x_0 + c_{\mathcal{N}}^T z \mid z \in \mathbb{R}^{n-1}\} \tag{3.2}$$

where $c_{\mathcal{N}}^T$ denotes a matrix whose columns form a basis of $\mathcal{N}(c^T)$, i.e., the null space of c^T.

A hyperplane divides $\mathbb{R}^n$ into two open halfspaces

$$\mathcal{H}^- = \{x \in \mathbb{R}^n \mid c^T x - d < 0\} \quad , \quad \mathcal{H}^+ = \{x \in \mathbb{R}^n \mid c^T x - d > 0\} \tag{3.3}$$

A closed halfspace is the closure of an open halfspace, e.g., $\overline{\mathcal{H}}^+ = \mathcal{H}^+ \cup \partial\mathcal{H} = \{x \in \mathbb{R}^n \mid c^T x \geq d\}$.

We will also use $c^T x = d$ to represent a hyperplane where $d = c^T x_0$ is a real scalar. We can rewrite this in the form (3.1) by finding a point in the plane $\mathcal{H}$, e.g., $x_0 = (c^T)^\dagger d$, where $(c^T)^\dagger$ denotes the Moore-Penrose inverse of the matrix c^T.

The intersection of a finite number of hyperplanes forms a so called affine set.

Definition 3.2 (Affine Sets)
An affine set is a set of the form

$$\mathcal{A} = \{x \in \mathbb{R}^n \mid C(x - x_0) = 0\} \tag{3.4}$$

where $C \in \mathbb{R}^{m \times n}$ and $x_0 \in \mathbb{R}^n$. Equivalently,

$$\mathcal{A} = \{x = x_0 + C_{\mathcal{N}} z \mid z \in \mathbb{R}^{n-m}\} \tag{3.5}$$

If the affine set is represented by $Cx = d$, where $d = Cx_0 \in \mathbb{R}^n$, we can choose $x_0 = C^\dagger d$ to convert to the representation (3.4).

Definition 3.3 (Polytopes and Polyhedra)
The set $\mathcal{P} \subseteq \mathbb{R}^n$ defined by an intersection of a finite number of halfspaces

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid c_i^T x - d_i > 0\} \tag{3.6}$$

where $1 \leq i \leq r$, is called a polyhedron. A bounded polyhedron is called a polytope.

Note that a polyhedron can be defined in terms of the formula $P : \mathbb{R}^n \to \mathbb{B}$,

$$P(x) = \bigwedge_{1 \le i \le r} [c_i^T x - d_i > 0] \tag{3.7}$$

Similarly, we can define the closure of $\mathcal{P}$ using

$$\overline{P}(x) = \bigwedge_{1 \le i \le r} [c_i^T x - d_i \ge 0] \tag{3.8}$$

We will only utilize some basic properties of polytopes and polyhedra; for an extensive treatment see, for instance, [64].

3.1.2 Quadratic Sets

We continue our treatment of relations over the reals by defining relations that can be described by quadratic equations and inequalities. That is, the relations can be expressed using formulas $Q : \mathbb{R}^n \to \mathbb{B}$ constructed, using logical connectives, from atomic formulas involving quadratic functions, $[q(x)\ \rho\ 0]$, with $q(x) = x^T Ax + 2b^T x + c$ where $x \in \mathbb{R}^n$, $A = A^T \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$, and $\rho \in \{=, \le, \ge, <, >, \ne\}$. We refer to such a relation as a semiquadratic set (cf. semialgebraic set).

Some familiar and useful sets can be constructed using one quadratic function. The general form of such a set is as follows.

Definition 3.4 (Quadratic Sets)
A quadratic set is a set of the form

$$\begin{aligned} \mathcal{Q} &= \{x \in \mathbb{R}^n \mid x^T Ax + 2b^T x + c \le 0\} && (3.9) \\ &= \{x \in \mathbb{R}^n \mid \begin{pmatrix} x \\ 1 \end{pmatrix}^T \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \le 0\} && (3.10) \end{aligned}$$

where $A = A^T \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$.

The intersection of an affine and quadratic set can be represented as follows

$$\mathcal{A} \cap \mathcal{Q} = \{x \in \mathbb{R}^n \mid x^T Ax + 2b^T x + c \le 0 \wedge Cx = d\} \tag{3.11}$$

or, using a constrained parameter representation,

$$\begin{aligned} \mathcal{A} \cap \mathcal{Q} = \{x = C^\dagger d + C_{\mathcal{N}} z \mid \\ \begin{pmatrix} z \\ 1 \end{pmatrix}^T \begin{pmatrix} C_{\mathcal{N}} & C^\dagger d \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \begin{pmatrix} C_{\mathcal{N}} & C^\dagger d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z \\ 1 \end{pmatrix} \le 0\}. \end{aligned} \tag{3.12}$$

We will now introduce some specific instances of quadratic sets. These are ellipsoids, cones, paraboloids and cylinders.

3.1.3 Ellipsoids

We begin our survey of quadratic sets by considering ellipsoids which are naturally parameterized by a shape matrix and an offset vector.

Definition 3.5 (Ellipsoids)
An ellipsoid centered at x_c is a set of the form

$$\mathcal{E} = \{x \in \mathbb{R}^n \mid (x - x_c)^T P (x - x_c) \leq 1\} \tag{3.13}$$

where the shape matrix $P^T = P$ is a positive definite matrix, which we denote by $P \succ 0$. This representation is unique, i.e., P and x_c are uniquely determined by the ellipsoid $\mathcal{E}$.

Equivalently, an ellipsoid can be described as a general quadratic set

$$\mathcal{E} = \{x \in \mathbb{R}^n \mid x^T A x + 2b^T x + c \leq 0\} \tag{3.14}$$

where $A = A^T \succ 0$ and $b^T A^{-1} b - c > 0$. The latter condition ensures that the ellipsoid does not reduce to a single point or is empty. This description is not unique; every other representation of $\mathcal{E}$ can be formed by scaling A, b and c by a positive factor.

We can always obtain a representation (3.14) from (3.13) by setting $A = P$, $b^T = -Px_c$ and $c = x_c^T P x_c - 1$. Similarly, given a description of an ellipsoid of the form (3.14), we can obtain the unique description (3.13) by setting $P = A/(b^T A^{-1} b - c)$ and $x_c = -A^{-1} b$.

The volume of an ellipsoid is given by

$$\text{vol}(\mathcal{E}) = \nu_n / \sqrt{\det(P)} \tag{3.15}$$

where ν_n is the volume of the unit sphere in $\mathbb{R}^n$. We will in Chapter 10 need to minimize the volume of ellipsoids under certain constraints. Hence, a useful observation is that the logarithm of the volume of an ellipsoid,

$$\log(\text{vol}(\mathcal{E})) = \log \nu_n + \frac{1}{2} \log \det(P^{-1}) \tag{3.16}$$

is a convex function of P. This enables us to perform optimization in an efficient manner.

3.1.4 Cones

An elliptic cone consists of two unbounded convex components, connected by a single point, the cone vertex.

Definition 3.6 (Cones)
An elliptic cone with vertex at x_c is a set of the form

$$\mathcal{CO} = \{x \in \mathbb{R}^n \mid (x - x_c)^T P (x - x_c) \leq 0\} \tag{3.17}$$

where $P = P^T \in \mathbb{R}^n$ has $n-1$ positive eigenvalues and one negative eigenvalue. Equivalently, after an affine change of coordinates, we can write

$$\mathcal{CO} = \{\begin{pmatrix} \tilde{z} \\ z_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{z}^T \tilde{P} \tilde{z} \leq z_n^2\} \tag{3.18}$$

where $\tilde{P} \in \mathbb{R}^{n-1 \times n-1}$, $\tilde{P} = \tilde{P}^T \succ 0$. That is, for each fixed value of the last coordinate z_n, we obtain an ellipsoid in the affine set formed by translating the subspace $\mathbb{R}^{n-1 \times n-1}$ by z_n^2.

We are only interested in one of the components along with the vertex, namely the component containing the positive z_n axis in representation (3.18). This translates to the part of the cone where $z_n \geq 0$, i.e.,

$$\mathcal{CO} = \{\begin{pmatrix} \tilde{z} \\ z_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{z}^T \tilde{P} \tilde{z} \leq z_n^2, \quad z_n \geq 0\} \tag{3.19}$$

We still refer to such sets simply as cones.

Since cones are unbounded sets, they have infinite volume. However, we can use the volume of the ellipsoid formed by setting $z_n = 1$ as a measure of the size of the cone.

Remark 3.1
The choice $z_n = 1$ is arbitrary. However, since we will be interested only in selecting the smallest cone fulfilling certain constraints, and not the actual volume of the resulting ellipsoid, the choice of z_n is of no importance.

3.1.5 Paraboloids

An elliptic paraboloid is an unbounded convex set.

Definition 3.7 (Paraboloids)
An elliptic paraboloid with base point at x_c is a set of the form

$$\mathcal{PB} = \{x \in \mathbb{R}^n \mid (x - x_c)^T P (x - x_c) \leq q^T (x - x_c)\} \tag{3.20}$$

where $P = P^T \in \mathbb{R}^n$ has $n-1$ positive eigenvalues and one eigenvalue that is zero. Equivalently, after an affine change of coordinates, we can write

$$\mathcal{PB} = \{\begin{pmatrix} \tilde{z} \\ z_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{z}^T \tilde{P} \tilde{z} \leq z_n\} \tag{3.21}$$

where $\tilde{P} \in \mathbb{R}^{n-1 \times n-1}$, $\tilde{P} = \tilde{P}^T \succ 0$. That is, for each fixed value of the last coordinate z_n, we obtain an ellipsoid in the affine set formed by translating the subspace $\mathbb{R}^{n-1 \times n-1}$ by z_n.

Analogously to the case of an elliptic cone, the representation (3.21) describes a set containing the whole z_n axis and having infinite volume. Again, we can use the volume of the ellipsoid formed by setting $z_n = 1$ as a measure of the size of the paraboloid.

3.1.6 Cylinders

The final quadratic set considered in this section is the elliptic cylinder.

Definition 3.8 (Cylinders)
An elliptic cylinder is a set of the form

$$\mathcal{CY} = \{x \in \mathbb{R}^n \mid x^T P x \leq 1\} \tag{3.22}$$

where $P = P^T \in \mathbb{R}^n$ has $n-1$ positive eigenvalues and one eigenvalue that is zero. Equivalently, after an affine change of coordinates, we can write

$$\mathcal{CY} = \{\begin{pmatrix} \tilde{z} \\ z_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{z}^T \tilde{P} \tilde{z} \leq 1\} \tag{3.23}$$

where $\tilde{P} \in \mathbb{R}^{n-1 \times n-1}, \tilde{P} = \tilde{P}^T \succ 0$. That is, a cylinder is formed as the union of ellipsoids $\{\tilde{z} \in \mathbb{R}^{n-1 \times n-1} \mid \tilde{z}^T \tilde{P} \tilde{z} \leq 1\}$ translated along the z_n axis.

As a measure of the size of cylinders, we use the volume of the $n-1$ dimensional ellipsoids.

Let us now examine closer how we perform computations with affine and quadratic sets. This is treated in the remainder of this chapter.

3.2 Feasibility

We are interested in determining if a set defined by a number of affine inequalities or by a quadratic inequality is empty, i.e., whether the set is feasible.

For affine inequalities, the feasibility problem is easily solved by linear programming [32]. For a quadratic inequality there exists an explicit test in terms of the parameters A, b and c if the parameterization $x^T Ax + 2b^T x + c \leq 0$ is used.

Lemma 3.1
Let $A^T = A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then the quadratic inequality

$$x^T Ax + 2b^T x + c \leq 0 \tag{3.24}$$

has no solution $x \in \mathbb{R}^n$ if and only if

$$A \succeq 0, \quad (I - AA^\dagger)b = 0, \quad \text{and} \quad b^T A^\dagger b - z < 0. \tag{3.25}$$

Proof See [32]. □

Sometimes a quadratic inequality does not impose any constraints on the variables, i.e., the set of solutions is the whole $\mathbb{R}^n$. Here follows the converse of Lemma 3.1.

Lemma 3.2
Let $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$. Then

$$\begin{pmatrix} x \\ 1 \end{pmatrix}^T \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \leq 0, \quad \forall x \in \mathbb{R}^n \quad \Leftrightarrow \quad \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \preceq 0 \tag{3.26}$$

Proof See [32]. □

If we have a system of both linear equations and a quadratic inequality it can always be reduce to a single quadratic inequality according to the equivalence between (3.11) and (3.12). Then Lemma 3.1 can be applied to determine feasibility. Alternatively, Lemma 3.2 can be used to check if the quadratic inequality does not impose any constraints.

3.3 Set Inclusion

Given sets defined by affine and quadratic inequalities we want to decide if one set is a subset of another. Here we will list a number of results on this topic.

3.3.1 The S-procedure

Given quadratic sets $\mathcal{F}_i = \{x \in \mathbb{R}^n \mid F_i(x) \leq 0\}$ where $F_i(x) = x^T A_i x + 2b_i^T x + c_i$, $A_i^T = A_i$ and $i = 0, \dots, p$ we want to check if one of the sets contains the intersection of the remaining sets. This can be formulated as the condition that one of the indicator functions should be negative while all the other indicator functions are negative. That is

$$\begin{gathered} \mathcal{F}_0 \supseteq \bigcap_{i=1}^{p} \mathcal{F}_i \\ \Updownarrow \\ F_0(x) \leq 0 \quad \forall x \in \mathbb{R}^n \text{ such that } F_i(x) \leq 0, \quad i = 1, \dots, p. \end{gathered} \tag{3.27}$$

The condition on the indicator functions can be formulated as an LMI, this reformulation is called the S-procedure.

Lemma 3.3
Let $F_0, \dots, F_p$ be quadratic functions of $x \in \mathbb{R}^n$, i.e.,

$$F_i(x) = x^T A_i x + 2b_i^T x + c_i \tag{3.28}$$

where $A_i^T = A_i$ and $i = 0, \dots, p$. Consider the following condition

$$F_0(x) \leq 0 \quad \forall x \in \mathbb{R}^n \textit{ such that } F_i(x) \leq 0, \quad i = 1, \dots, p. \tag{3.29}$$

A sufficient condition for (3.29) to hold is

$$\exists \tau_1 \geq 0, \dots, \tau_p \geq 0 \text{ such that } \forall x \in \mathbb{R}^n,\ F_0(x) \leq \sum_{i=1}^{p} \tau_i F_i(x) \tag{3.30}$$

If $p = 1$ and there exists an x_0 such that $F_1(x_0) < 0$ then condition (3.30) is also necessary.

Proof See [61]. □

According to Lemma 3.2 condition (3.30) can be reformulated as

$$\begin{gathered}\begin{pmatrix} A_0 & b_0 \\ b_0^T & c_0 \end{pmatrix} - \sum_{i=1}^{p} \tau_i \begin{pmatrix} A_i & b_i \\ b_i^T & c_i \end{pmatrix} \preceq 0 \\ \tau_i \geq 0, \quad i = 1, \dots, p \end{gathered} \tag{3.31}$$

which is an LMI in the τ_i's

3.3.2 Polytope in Quadratic Set

The first inclusion we consider is the one of a polytope in a quadratic set, that is, formulate conditions to determine if $\mathcal{P} \subseteq \mathcal{Q}$. The simplest case is when the polyhedron is given as the convex hull of points, $\mathcal{P} = \mathbf{Co}\{v_1, \dots, v_p\}$ and the quadratic set is convex. Then it suffices to check if all points belong to the set, i.e.,

$$v_i^T A v_i + 2b^T v_i + c \leq 0, \quad i = 1, \dots, p. \tag{3.32}$$

In case the polytope is given as the intersection of halfspaces, $\mathcal{P} = \{x \in \mathbb{R}^n \mid c_i^T x \leq d_i,\ i = 1, \dots, p\}$, the situation is somewhat more complex. If the quadratic set is convex, we can compute the vertices of the polytope and use equation (3.32) to check for inclusion. The alternative is to use the S-procedure with $F_0(x)$ quadratic and the $F_i(x)$'s affine, i.e., solve the LMI

$$\begin{gathered}\begin{pmatrix} A & b \\ b^T & c \end{pmatrix} - \sum_{i=1}^{p} \tau_i \begin{pmatrix} 0 & c_i/2 \\ c_i^T/2 & -d_i \end{pmatrix} \preceq 0 \\ \tau_i \geq 0, \quad i = 1, \dots, p. \end{gathered} \tag{3.33}$$

The only drawback is that this condition is only sufficient for inclusion. This means that we may not find a solution to (3.33) although the polytope actually is a subset of the quadratic set. On the other hand, we do not have to restrict ourselves to convex quadratic sets, and for that matter, we may as well treat polyhedra instead of polytopes.

3.3.3 Quadratic and Affine Set in Quadratic Set

To determine if the intersection of a quadratic set and an affine set is a subset of another quadratic set we can utilize the S-procedure. We have the following lemma [32].

Lemma 3.4
Consider the quadratic sets $\mathcal{Q}_i = \{x \in \mathbb{R}^n \mid x^T A_i x + 2b_i^T x + c_i \leq 0\}, i = 1, 2$ and the affine set $\mathcal{A} = \{x \in \mathbb{R}^n \mid Cx = d\}$. Then $\mathcal{Q}_1 \cap \mathcal{A} \subseteq \mathcal{Q}_2$ if and only if there exists $\tau \in \mathbb{R}$ such that

$$M^T(\begin{pmatrix} A_2 & b_2 \\ b_2^T & c_2 \end{pmatrix} - \tau \begin{pmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{pmatrix})M \preceq 0, \quad \tau \geq 0 \tag{3.34}$$

where

$$M = \begin{pmatrix} C_{\mathcal{N}} & C^\dagger d \\ 0 & 1 \end{pmatrix}. \tag{3.35}$$

Proof First, note that $\mathcal{Q}_1 \cap \mathcal{A} \subseteq \mathcal{Q}_2$ is equivalent to $\mathcal{Q}_1 \cap \mathcal{A} \subseteq \mathcal{Q}_2 \cap \mathcal{A}$. The intersection $\mathcal{Q}_i \cap \mathcal{A}$ can be represented as in (3.11), i.e.,

$$\mathcal{Q}_i \cap \mathcal{A} = \{x = Cz + C^\dagger d \mid \hat{z}^T M^T \begin{pmatrix} A_i & b_i \\ b_i^T & c_i \end{pmatrix} Mz \leq 0\} \tag{3.36}$$

where $\hat{z} = (z, 1)^T$. To determine if $\mathcal{Q}_1 \cap \mathcal{A} \subseteq \mathcal{Q}_2 \cap \mathcal{A}$ is true we only have to apply the S-procedure to the inequality constraints of (3.36) for $i = 1, 2$, since subset relations are preserved under affine mappings. □

In addition, we will want to formulate conditions that enable us to check if a set formed as the intersection of a quadratic set, an affine set and a number of halfspaces is contained in a quadratic set.

Lemma 3.5
Consider the quadratic sets $\mathcal{Q}_i = \{x \in \mathbb{R}^n \mid x^T A_i x + 2b_i^T x + c_i \leq 0\}, i = 1, 2$, the affine set $\mathcal{A} = \{x \in \mathbb{R}^n \mid Cx = d\}$ and the m halfspaces $\mathcal{H}_j = \{x \in \mathbb{R}^n \mid c_j \leq d_j\}, j = 2, \ldots, m+1$. Then

$$\mathcal{Q}_1 \cap \mathcal{A} \cap \bigcap_{2 \leq j \leq m+1} \mathcal{H}_j \subseteq \mathcal{Q}_2 \tag{3.37}$$

if there exist $\tau_1, \tau_j \in \mathbb{R}$, $\tau_1, \tau_j \geq 0$ such that

$$M^T(\begin{pmatrix} A_2 & b_2 \\ b_2^T & c_2 \end{pmatrix} - \tau_1 \begin{pmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{pmatrix} - \sum_{j=2}^{m+1} \tau_j \begin{pmatrix} 0 & c_j/2 \\ c_j/2 & -d_j \end{pmatrix})M \preceq 0 \tag{3.38}$$

where

$$M = \begin{pmatrix} C_{\mathcal{N}} & C^\dagger d \\ 0 & 1 \end{pmatrix}. \tag{3.39}$$

Proof The same arguments as in the proof of Lemma 3.4 and application of the S-procedure provide the sufficient conditions. $\square$

Again, a potential problem with the formulation of Lemma 3.5 is that the S-procedure only provides us with sufficient conditions for the inclusion. Depending on the amount of conservatism introduced we may conclude that the inclusion is false even when it is not.

4

Power Cones

In addition to quadratic and affine sets, we utilize sets defined by constraints that are quadratic in $n-1$ variables and have a special kind of nonlinearity in one variable, namely the power function. Hence, we refer to such sets as power cones. In this chapter, we discuss how such sets can be represented and operated on. Due to the nonlinearity in one variable, we utilize quadratic and affine approximations which facilitate efficient representation and manipulations. These approximations are also described.

4.1 Representation

We consider sets that have an indicator function which is quadratic in all variables except x_n where it is the power function, $f : \mathbb{R}^+ \to \mathbb{R}^+,\ f(x) = x^r,\ r \in \mathbb{R}$, see Figure 4.1.

Definition 4.1 (Power Cones)
A power cone is a set of the form

$$\mathcal{PC} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r\ ,\ x_n \geq 0\} \tag{4.1}$$

where $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0,\ \tilde{x} \in \mathbb{R}^{n-1}$ *and* $x_n,\ r \in \mathbb{R}$.

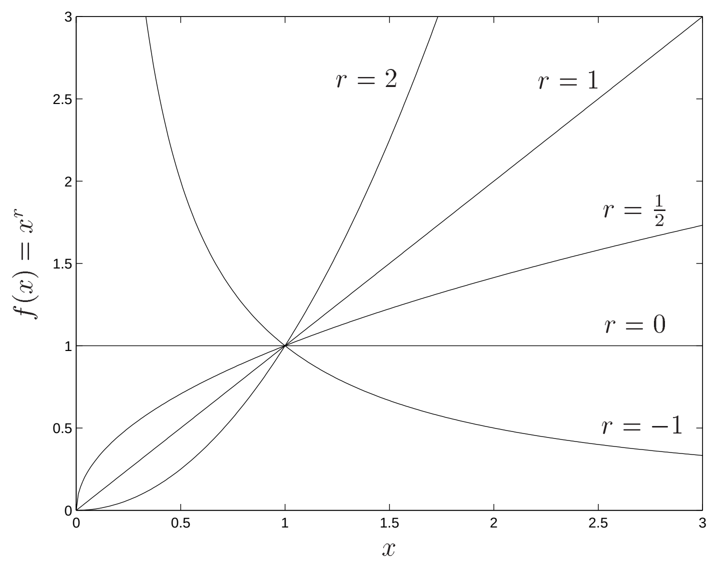


Figure 4.1: The power function for different values of r.

Note that we can form cones, paraboloids and cylinders as special instances of power cones by selecting r as $2, 1$, and 0, respectively.

Example 4.1 *Consider the family of power cones in $\mathbb{R}^3$ parameterized by r and described by*

$$\mathcal{PC} = \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \mid \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 10 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq x_3^r \} \tag{4.2}$$

$$= \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \mid 10x_1^2 + 2x_1x_2 + x_2^2 \leq x_3^r \} \tag{4.3}$$

In Figure 4.2 we plot the sets resulting from four different choices of the power parameter r. The set corresponding to $r = 0$ and $r = 1$ is a cylinder and a paraboloid, respectively, while the other two sets cannot be described using quadratic functions.

In Example 4.1 we see some typical features of power cones. The four different basic shapes of Figure 4.2 are representative for the shapes that can be obtained with different selections of the power parameter r. We can characterize these different shapes, in terms of convexity and the volume of the ellipsoid $\mathcal{E}_{x_n}$ formed by intersecting $\mathcal{PC}$ with a hyperplane perpendicular to the x_n axis, as follows.

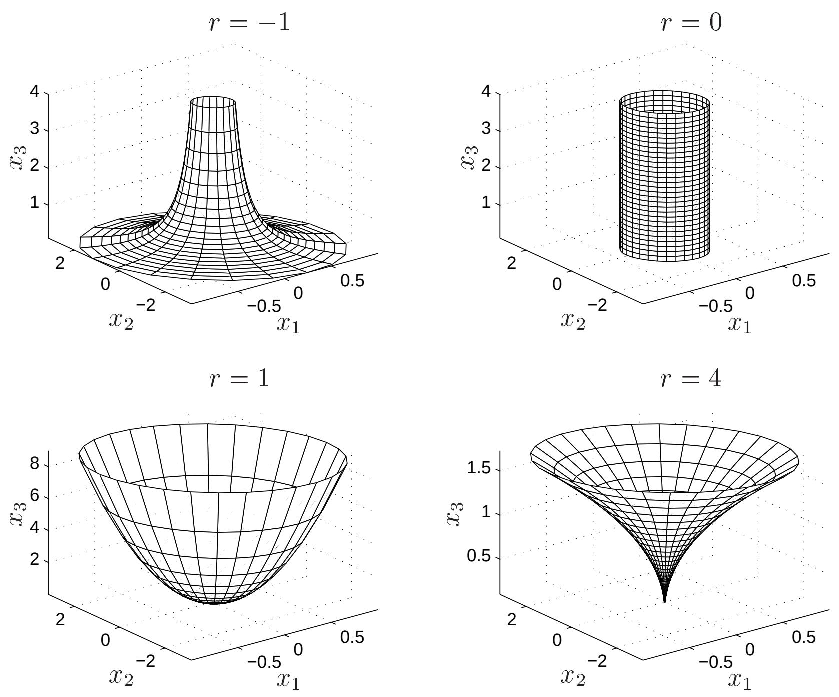


Figure 4.2: Power cones for different values of r.

- $r \in (-\infty, 0)$: Non-convex set, $\mathrm{vol}(\mathcal{E}_{x_n})$ is a monotonically decreasing function of x_n.
- $r = 0$: Convex set, $\mathrm{vol}(\mathcal{E}_{x_n})$ is constant w.r.t. x_n.
- $r \in (0, 2]$: Convex set, $\mathrm{vol}(\mathcal{E}_{x_n})$ increases monotonically with x_n.
- $r \in (2, \infty)$: Non-convex set, $\mathrm{vol}(\mathcal{E}_{x_n})$ increases monotonically with x_n.

That is, the variation in volume with x_n depends on whether r is less, equal or greater than zero, while convexity is confined to r in the interval $[0, 2]$, see Figure 4.3.

In Figure 4.1 we observe the same basic shape categories although upper bound of r for convexity differs. From Figure 4.1 we see that the sets defined by $y \leq x^r$ are convex for $r \in [0, 1]$. Since the power cones are defined in terms of a quadratic function in $\tilde{x}$, the interval for convexity is $r \in [0, 2]$, cf. the convexity of sets defined by $y^2 \leq x^r$.

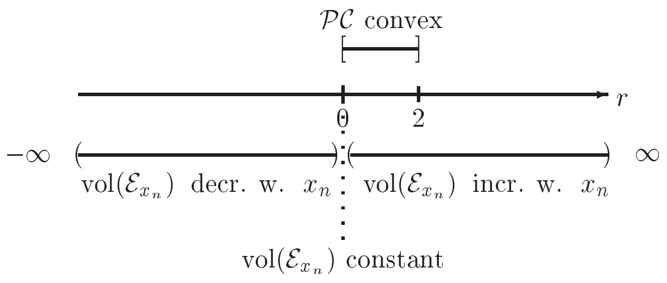

$\mathcal{PC}$ convex
r
0
2
$-\infty$
∞
vol($\mathcal{E}_{x_n}$) decr. w. x_n
vol($\mathcal{E}_{x_n}$) incr. w. x_n
vol($\mathcal{E}_{x_n}$) constant

Figure 4.3: Variation of $\mathcal{PC}$ properties with r.

4.2 The Intersection $\mathcal{PC} \cap \mathcal{A}$

In Section 3.1.2 we saw that we can represent the intersection of a quadratic set and an affine set using a single quadratic inequality, (3.12). In the case of the intersection of a power cone and an affine set, we may of course write down the result using the constraint representation

$$\mathcal{PC} \cap \mathcal{A} = \{x = \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r \wedge Cx = d\} \tag{4.4}$$

However, for checking feasibility and set inclusion, we would prefer a description similar to (3.12). But since the power cones are nonlinear in one variable, this can not be done. Instead, we compute quadratic approximations which can then be represented using (3.12). Both outer and inner approximations are needed. They are described in the next two sections.

4.2.1 Outer Approximation

We resolve the problem of finding a conservative outer approximation of the intersection $\mathcal{PC} \cap \mathcal{A}$ by computing the smallest possible set which is guaranteed to contain $\mathcal{PC} \cap \mathcal{A}$ and allows a convenient representation. The basic idea is to first find a set, $\mathcal{Q} \subseteq \mathbb{R}^n$, (either a halfspace, cone, paraboloid or a cylinder) such that $\mathcal{Q} \supseteq \mathcal{PC} \cap \mathcal{A}$. We then compute the intersection $\mathcal{Q} \cap \mathcal{A}$ and, since subset relations are preserved under affine mappings, we can use the result as an outer approximation of $\mathcal{PC} \cap \mathcal{A}$.

We thus need to find a way to compute the set $\mathcal{Q}$ in such a way that we can guarantee that $\mathcal{Q} \supseteq \mathcal{PC} \cap \mathcal{A}$. This involves selecting the type of set to use and is dependent on the value of the power parameter r. This implies that we have to distinguish between several cases. It turns out that these are

1) $r \in (0, 1)$

2) $r \in (1, 2)$

3) $r \in (2, \infty)$

4) $r \in (-\infty, 0)$

These four cases are treated separately below.

Case 1: $r \in (0, 1)$**.**

Consider Figure 4.4. Excluding the case when $\mathcal{A}$ is parallel to the x_n axis or lies in the subspace perpendicular to it, the first observation is that there will always be two values $\underline{x}_n$ and $\overline{x}_n$ such that $\mathcal{PC} \cap \mathcal{A}$ is a subset of $\mathcal{PC}$ only for $\underline{x}_n \le x_n \le \overline{x}_n$. Each constant value of x_n defines an ellipsoid $\mathcal{E}_{x_n}^{\mathcal{PC}}$ obtained by intersecting $\mathcal{PC}$

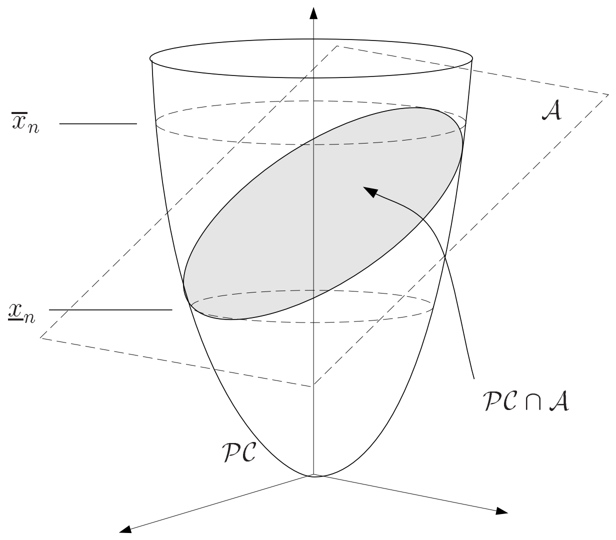


Figure 4.4: The intersection $\mathcal{PC} \cap \mathcal{A}$ when $r \in (0, 2)$.

with a hyperplane perpendicular to the x_n axis and intersecting it at x_n. We now aim at finding a quadratic set $\mathcal{Q}$ such that $\mathcal{E}_{x_n}^{\mathcal{PC}} \subseteq \mathcal{E}_{x_n}^{\mathcal{Q}}$ for each $x_n \in [\underline{x}_n, \overline{x}_n]$ where $\mathcal{E}_{x_n}^{\mathcal{Q}}$ is the ellipsoid formed by intersecting $\mathcal{Q}$ with the same hyperplane. This is easily accomplished by taking $\mathcal{Q}$ as the cylinder having $\mathcal{E}_{\overline{x}_n}$ as cross section, that is

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \le 1 \} \tag{4.5}$$

where $\alpha = 1/\overline{x}_n^r$. The intersection of this cylinder with $\mathcal{A}$ is then guaranteed to contain $\mathcal{PC} \cap \mathcal{A}$.

Case 2: $r \in (1, 2)$.

In this case we use the same reasoning as for $r \in (0, 1)$ since here $\mathcal{PC}$ basically has the same shape. The difference is that we can do better than just taking $\mathcal{Q}$ as the cylinder having the largest ellipsoid as cross section. Instead we use a paraboloid having both $\mathcal{E}_{\underline{x}_n}$ and $\mathcal{E}_{\overline{x}_n}$ as cross sections. Since we know that the powercone connects these two ellipsoids at a rate that is slower than for a paraboloid, we know that $\mathcal{E}_{x_n}^{\mathcal{PC}} \subseteq \mathcal{E}_{x_n}^{\mathcal{Q}}$ for each $x_n \in [\underline{x}_n, \overline{x}_n]$.

We consider a paraboloid obtained using a scaled version of P and having its base point translated along the x_n axis. That is

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T (\alpha P) \tilde{x} \leq x_n - x_{n,0} \} \tag{4.6}$$

where $\alpha, x_{n,0}$ are positive scalars. The requirement that both $\mathcal{Q}$ and $\mathcal{PC}$ have the same ellipsoids as cross sections for $x_n = \underline{x}_n$ and $\overline{x}_n$ leads to the equations

$$\underline{x}_n - x_{n,0} = \underline{x}_n^r \alpha \tag{4.7}$$
$$\overline{x}_n - x_{n,0} = \overline{x}_n^r \alpha \tag{4.8}$$

Solving for $x_{n,0}$ and α leads to

$$x_{n,0} = \frac{\overline{x}_n \underline{x}_n^r - \underline{x}_n \overline{x}_n^r}{\underline{x}_n^r - \overline{x}_n^r} \tag{4.9}$$
$$\alpha = \frac{\underline{x}_n - x_{n,0}}{\underline{x}_n^r} \tag{4.10}$$

and we thus have a quadratic set such that $\mathcal{Q} \cap \mathcal{A} \supseteq \mathcal{PC} \cap \mathcal{A}$.

Case 3: $r \in (2, \infty)$.

In this interval $\mathcal{PC}$ is nonconvex and the volume of $\mathcal{E}_{x_n}$ increases monotonically in x_n, see Figure 4.5. Excluding again the case when $\mathcal{A}$ is parallel to the x_n axis, we see that three different cases may arise. First and simplest is the case when $\mathcal{A}$ is perpendicular to the x_n axis, resulting in $\mathcal{PC} \cap \mathcal{A}$ being quadratic and thus easily represented. The second case arises when $\mathcal{A}$ is aligned in such a way that the intersection with $\mathcal{PC}$ is a connected unbounded set. The third case is when the alignment of $\mathcal{A}$ leads to an intersection that consists of two unconnected components, one of them bounded and the other unbounded as shown in Figure 4.5.

Example 4.2 *Consider the power cone*

$$\mathcal{PC} = \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \mid \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 2 & 1/2 \\ 1/2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq x_3^4 \} \tag{4.11}$$

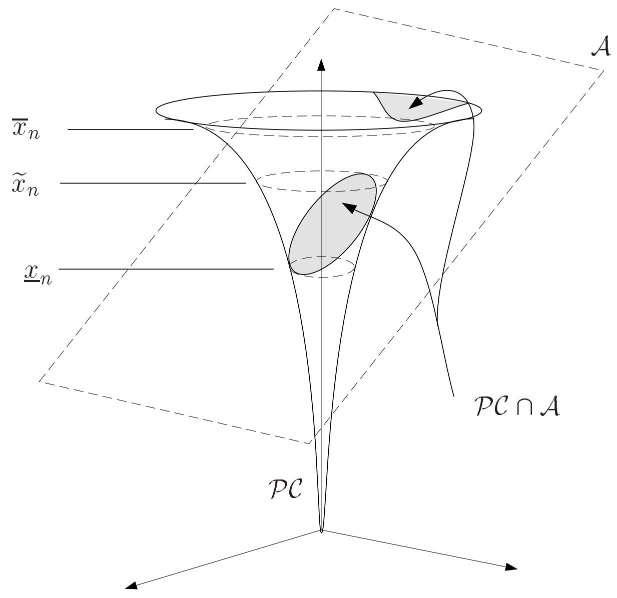


Figure 4.5: The intersection $\mathcal{PC} \cap \mathcal{A}$ when $r \in (2, \infty)$.

In Figure 4.6 we illustrate the three different cases obtained when $\mathcal{PC}$ is intersected with hyperplanes

$$\mathcal{A}_i = \{x \in \mathbb{R}^n \mid c_i^T x = d_i\} \quad , \quad i = 1, 2, 3 \tag{4.12}$$

where $c_1 = (0, 0, 1)^T$, $c_2 = (-1, 0, 1)^T$, $c_3 = (-0.5, 0, 1)^T$ and $d_1 = 0.5$, $d_2 = 4$, $d_3 = 0.5$.

We thus have to treat these cases separately. First, we need to be able to distinguish between them. This is treated in Section 4.2.3. Second, we need to decide how to approximate $\mathcal{PC} \cap \mathcal{A}$ in the two latter cases where it is not quadratic.

When the intersection is a single connected component, we can find a value of x_n, say $\underline{x}_n$, such that $\mathcal{PC} \cap \mathcal{A}$ is contained in the part of $\mathcal{PC}$ where $x_n \geq \underline{x}_n$. Hence, it is also contained in the halfspace

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid x_n \geq \underline{x}_n \} \tag{4.13}$$

and we can use the set $\mathcal{Q} \cap \mathcal{A}$ as an approximation of the intersection.

If the intersection consists of two unconnected components, there are three values, $\underline{x}_n, \widetilde{x}_n$ and $\overline{x}_n$ such that one component is contained in the part of $\mathcal{PC}$ where $x_n \geq \overline{x}_n$ and the other is contained in the section where $\underline{x}_n \leq x_n \leq \widetilde{x}_n$. The

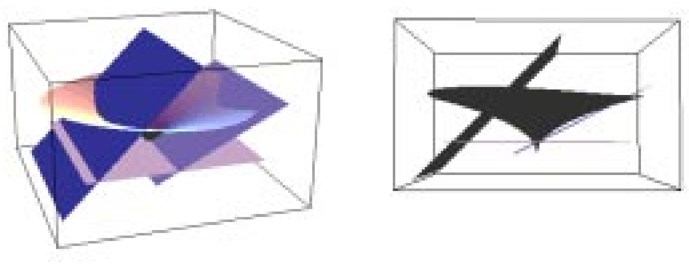


Figure 4.6: The three different cases arising when intersecting the power cone with a hyperplane.

former will be a subset of the set $\mathcal{Q}_1 \cap \mathcal{A}$ where

$$\mathcal{Q}_1 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid x_n \geq \overline{x}_n \} \tag{4.14}$$

while the other will lie in a cone having $\mathcal{E}_{\widetilde{x}_n}$ and $\mathcal{E}_{\underline{x}_n}$ as cross sections.

Similarly to case 2, we intend to use a cone having a scaled version of P as its shape matrix and its vertex translated along the x_n axis,

$$\mathcal{Q}_2 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.15}$$

where $\alpha, x_{n,0}$ are positive scalars. The requirement that both $\mathcal{Q}$ and $\mathcal{PC}$ have the same ellipsoids as cross sections for $x_n = \underline{x}_n$ and $\widetilde{x}_n$ leads to the equations

$$(\underline{x}_n - x_{n,0})^2 = \underline{x}_n^r \alpha \tag{4.16}$$
$$(\widetilde{x}_n - x_{n,0})^2 = \widetilde{x}_n^r \alpha \tag{4.17}$$

and solving for $x_{n,0}$ and α gives

$$x_{n,0} = \frac{\widetilde{x}_n \underline{x}_n^{r/2} - \underline{x}_n \widetilde{x}_n^{r/2}}{\underline{x}_n^{r/2} - \widetilde{x}_n^{r/2}} \tag{4.18}$$
$$\alpha = \frac{(\underline{x}_n - x_{n,0})^2}{\underline{x}_n^r} \tag{4.19}$$

and hence we have a cone $\mathcal{Q}_2$ such that the bounded component of the intersection lies in $\mathcal{Q}_2 \cap \mathcal{A}$.

Case 4: $r \in (-\infty, 0)$.

Here we basically have the same situation as in case 3, that is the set $\mathcal{PC}$ has the same basic shape except that it is turned upside down. Hence, in the case of one component, we can approximate the intersection in the same manner using the set

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid x_n \leq \underline{x}_n \}, \tag{4.20}$$

since $\mathcal{PC} \cap \mathcal{A}$ is known to lie in the part of the power cone below $\underline{x}_n$. Similarly, in the case of two unconnected components, we can approximate them in a fashion analog to that of case 3 using the set

$$\mathcal{Q}_1 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid x_n \leq \underline{x}_n \}, \tag{4.21}$$

and the set

$$\mathcal{Q}_2 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T (\alpha P) \tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.22}$$

where $x_{n,0}$ and α can be computed as

$$x_{n,0} = \frac{\widetilde{x}_n \overline{x}_n^{r/2} - \overline{x}_n \widetilde{x}_n^{r/2}}{\overline{x}_n^{r/2} - \widetilde{x}_n^{r/2}} \tag{4.23}$$

$$\alpha = \frac{(\overline{x}_n - x_{n,0})^2}{\overline{x}_n^r} \tag{4.24}$$

Summary

For sake of clarity, we summarize here the results of this section. Note that we have yet to specify how the critical values of x_n are computed. This will be treated in Section 4.2.3 where some of the conditions needed to distinguish between different cases can be found.

We wish to find a set $\mathcal{Q} \cap \mathcal{A}$ such that the subset relation

$$\mathcal{PC} \cap \mathcal{A} \subseteq \mathcal{Q} \cap \mathcal{A} \tag{4.25}$$

is guaranteed to hold. Here,

$$\mathcal{PC} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r \ , \ x_n \geq 0 \} \tag{4.26}$$

with $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and x_n, $r \in \mathbb{R}$, and

$$\mathcal{A} = \{ x \in \mathbb{R}^n \mid Cx = d \} \tag{4.27}$$

where $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^n$. We seek a set of the form

$$\mathcal{Q} \cap \mathcal{A} = (\mathcal{Q}_1 \cap \mathcal{A}) \cup (\mathcal{Q}_2 \cap \mathcal{A}) \tag{4.28}$$

where

$$\mathcal{Q}_1 \cap \mathcal{A} = \{x \in \mathbb{R}^n \mid Cx = d \wedge x_n \geq \beta\} \tag{4.29}$$

$$\mathcal{Q}_2 \cap \mathcal{A} = \{x = x_0 + C_{\mathcal{N}} z \mid \hat{z}^T M^T \hat{P} M \hat{z} \leq 0\} \tag{4.30}$$

and

$$\hat{z} = \begin{pmatrix} z \\ 1 \end{pmatrix} \quad , \quad M = \begin{pmatrix} C_{\mathcal{N}} & C^{\dagger} d \\ 0 & 1 \end{pmatrix} \tag{4.31}$$

Depending on the value of the power parameter r, we distinguish between four different cases.

- Case 1: $r \in (0, 1)$. Here, $\mathcal{Q}_1 = \emptyset$ and

$$\hat{P} = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/\overline{x}_n^r \tag{4.32}$$

- Case 2: $r \in (1, 2)$. $\mathcal{Q}_1 = \emptyset$ and

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & -1/2 \\ 0 & -1/2 & x_{n,0} \end{pmatrix}, \\ x_{n,0} &= \frac{\overline{x}_n \underline{x}_n^r - \underline{x}_n \overline{x}_n^r}{\underline{x}_n^r - \overline{x}_n^r} \quad , \quad \alpha = \frac{\underline{x}_n - x_{n,0}}{\underline{x}_n^r} \end{aligned} \tag{4.33}$$

- Case 3: $r \in (2, \infty)$. Here we need to distinguish between three cases, depending on the number of positive real solutions to (4.78).
  - 1 solution: Now, $\mathcal{Q}_2 = \emptyset$ and $\beta = \widetilde{x}_n$.
  - $\tilde{C} P^{-1} \tilde{C}^T = 0$: $\mathcal{Q}_1 = \emptyset$ and

$$\hat{P} = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/d^r \tag{4.34}$$

  - 3 solutions: $\beta = \overline{x}_n$ and

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{\widetilde{x}_n \underline{x}_n^{r/2} - \underline{x}_n \widetilde{x}_n^{r/2}}{\underline{x}_n^{r/2} - \widetilde{x}_n^{r/2}} \quad , \quad \alpha = \frac{(\underline{x}_n - x_{n,0})^2}{\underline{x}_n^r} \end{aligned} \tag{4.35}$$

- Case 4: $r \in (-\infty, 0)$. Here we have a representation analog to that for case 3 except for some minor changes. That is, we distinguish between three cases,

- 1 solution: Now, $\mathcal{Q}_2 = \emptyset$ and $\beta = \widetilde{x}_n$.
- $\tilde{C}P^{-1}\tilde{C}^T = 0$: $\mathcal{Q}_1 = \emptyset$ and

$$\hat{P} = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/d^r \tag{4.36}$$

- 3 solutions: $\beta = \underline{x}_n$ and

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{\widetilde{x}_n \overline{x}_n^{r/2} - \overline{x}_n \widetilde{x}_n^{r/2}}{\overline{x}_n^{r/2} - \widetilde{x}_n^{r/2}} \quad , \quad \alpha = \frac{(\overline{x}_n - x_{n,0})^2}{\overline{x}_n^r} \end{aligned} \tag{4.37}$$

4.2.2 Inner Approximation

The treatment of inner approximations goes along similar lines as the outer approximations. However, we need to use different sets in different situations, hence we go through the detailed treatment of each case and summarize at the end as before.

Case 1: $r \in (0, 1)$.

Consider again Figure 4.4. We know that there will always be two values $\underline{x}_n$ and $\overline{x}_n$ such that $\mathcal{PC} \cap \mathcal{A}$ is a subset of $\mathcal{PC}$ only for $\underline{x}_n \leq x_n \leq \overline{x}_n$, hence we focus on this section. Again, each constant value of x_n defines an ellipsoid $\mathcal{E}_{x_n}^{\mathcal{PC}}$ obtained by intersecting $\mathcal{PC}$ with a hyperplane perpendicular to the x_n axis and intersecting it at x_n. We now aim at finding a quadratic set $\mathcal{Q}$ such that $\mathcal{E}_{x_n}^{\mathcal{PC}} \supseteq \mathcal{E}_{x_n}^{\mathcal{Q}}$ for each $x_n \in [\underline{x}_n, \overline{x}_n]$ where $\mathcal{E}_{x_n}^{\mathcal{Q}}$ is the ellipsoid formed by intersecting $\mathcal{Q}$ with the same hyperplane. This is easily accomplished by taking $\mathcal{Q}$ as the paraboloid having $\mathcal{E}_{\underline{x}_n}$ and $\mathcal{E}_{\overline{x}_n}$ as its cross sections. We consider a paraboloid obtained using a scaled version of P with its base point translated along the x_n axis. That is

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T (\alpha P) \tilde{x} \leq x_n - x_{n,0} \} \tag{4.38}$$

where $\alpha, x_{n,0}$ are positive scalars. The requirement that both $\mathcal{Q}$ and $\mathcal{PC}$ have the same ellipsoids as cross sections for $x_n = \underline{x}_n$ and $\overline{x}_n$ leads to the equations

$$\underline{x}_n - x_{n,0} = \underline{x}_n^r \alpha \tag{4.39}$$

$$\overline{x}_n - x_{n,0} = \overline{x}_n^r \alpha \tag{4.40}$$

Solving for $x_{n,0}$ and α leads to

$$x_{n,0} = \frac{\overline{x}_n \underline{x}_n^r - \underline{x}_n \overline{x}_n^r}{\underline{x}_n^r - \overline{x}_n^r} \tag{4.41}$$

$$\alpha = \frac{\underline{x}_n - x_{n,0}}{\underline{x}_n^r} \tag{4.42}$$

and we thus have a quadratic set such that $\mathcal{Q} \cap \mathcal{A} \subseteq \mathcal{PC} \cap \mathcal{A}$.

Case 2: $r \in (1, 2)$**.**

Here, $\mathcal{PC}$ basically has the same shape as for $r \in (0, 1)$ but we now use a quadratic cone having both $\mathcal{E}_{\underline{x}_n}$ and $\mathcal{E}_{\overline{x}_n}$ as cross sections. We use a cone obtained using a scaled version of P and having its vertex translated along the x_n axis. That is

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0}\})^2 \tag{4.43}$$

where $\alpha, x_{n,0}$ are positive scalars. The requirement that both $\mathcal{Q}$ and $\mathcal{PC}$ have the same ellipsoids as cross sections for $x_n = \underline{x}_n$ and $\overline{x}_n$ leads to the equations

$$(\underline{x}_n - x_{n,0})^2 = \underline{x}_n^r \alpha \tag{4.44}$$

$$(\overline{x}_n - x_{n,0})^2 = \overline{x}_n^r \alpha \tag{4.45}$$

Solving for $x_{n,0}$ and α leads to

$$x_{n,0} = \frac{\overline{x}_n \underline{x}_n^{r/2} - \underline{x}_n \overline{x}_n^{r/2}}{\underline{x}_n^{r/2} - \overline{x}_n^{r/2}} \tag{4.46}$$

$$\alpha = \frac{(\underline{x}_n - x_{n,0})^2}{\underline{x}_n^r} \tag{4.47}$$

and we thus have a quadratic set such that $\mathcal{Q} \cap \mathcal{A} \subseteq \mathcal{PC} \cap \mathcal{A}$.

Case 3: $r \in (2, \infty)$**.**

In this interval $\mathcal{PC}$ is nonconvex and the volume of $\mathcal{E}_{x_n}$ increases monotonically in x_n as shown in Figure 4.5.

Again, three different cases may arise. The case when $\mathcal{A}$ is perpendicular to the x_n axis results in $\mathcal{PC} \cap \mathcal{A}$ being quadratic and hence is easily represented. The second case arises when $\mathcal{A}$ is aligned in such a way that the intersection with $\mathcal{PC}$ is a connected unbounded set. The third case is when the alignment of $\mathcal{A}$ leads to an intersection that consists of two unconnected components, one of them bounded and the other unbounded as shown in Figure 4.5.

When the intersection is a single connected component, we can find a value of x_n, say $\underline{x}_n$, such that $\mathcal{PC} \cap \mathcal{A}$ is contained in the part of $\mathcal{PC}$ where $x_n \geq \underline{x}_n$. Taking $\mathcal{Q}$ as a quadratic cone having the ellipsoid $\mathcal{E}_{\underline{x}_n}$ as its cross section and the same tangent planes as $\mathcal{PC}$ at $x_n = \underline{x}_n$ will, according to Lemma 4.2 of Section 4.3.1, guarantee that $\mathcal{Q} \subseteq \mathcal{PC}$ and hence that $\mathcal{Q} \cap \mathcal{A} \subseteq \mathcal{PC} \cap \mathcal{A}$. That is, we use the cone

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.48}$$

with $x_n \geq x_{n,0}$ and

$$x_{n,0} = \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}}. \tag{4.49}$$

If the intersection consists of two unconnected components, there are three values, $\underline{x}_n, \widetilde{x}_n$ and $\overline{x}_n$ such that one component is contained in the part of $\mathcal{PC}$ where $x_n \geq \overline{x}_n$ and the other is contained in the section where $\underline{x}_n \leq x_n \leq \widetilde{x}_n$. The former will be a subset of the set $\mathcal{Q}_1 \cap \mathcal{A}$ where

$$\mathcal{Q}_1 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.50}$$

with $x_n \geq x_{n,0}$ and

$$x_{n,0} = \frac{\overline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \overline{x}_n^{r-2}}. \tag{4.51}$$

while the other will lie in a cylinder having $\mathcal{E}_{\underline{x}_n}$ as cross section, that is

$$\mathcal{Q}_2 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq 1 \} \tag{4.52}$$

where $\alpha = 1/\underline{x}_n^r$. The intersection of this cylinder with $\mathcal{A}$ is then guaranteed to contain the bounded component of $\mathcal{PC} \cap \mathcal{A}$.

Case 4: $r \in (-\infty, 0)$**.**

Here we basically have the same situation as in case 3, that is the set $\mathcal{PC}$ has the same basic shape except that it is turned upside down. Hence we can, in the case of one component, approximate the intersection in the same manner using the set

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.53}$$

with $x_n \geq x_{n,0}$ and

$$x_{n,0} = \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}}. \tag{4.54}$$

and in the case of two components using the sets $\mathcal{Q}_1$ and $\mathcal{Q}_2$ where

$$\mathcal{Q}_1 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2 \} \tag{4.55}$$

with $x_n \leq x_{n,0}$ and

$$x_{n,0} = \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}}. \tag{4.56}$$

and

$$\mathcal{Q}_2 = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq 1 \} \tag{4.57}$$

where $\alpha = 1/\overline{x}_n^r$.

Summary

We wish to find a set $\mathcal{Q} \cap \mathcal{A}$ such that the subset relation

$$\mathcal{PC} \cap \mathcal{A} \supseteq \mathcal{Q} \cap \mathcal{A} \tag{4.58}$$

is guaranteed to hold. Here,

$$\mathcal{PC} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r \ , \ x_n \geq 0 \} \tag{4.59}$$

with $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and x_n, $r \in \mathbb{R}$, and

$$\mathcal{A} = \{ x \in \mathbb{R}^n \mid Cx = d \} \tag{4.60}$$

where $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^n$. We seek a set of the form

$$\mathcal{Q} \cap \mathcal{A} = (\mathcal{Q}_1 \cap \mathcal{A}) \cup (\mathcal{Q}_2 \cap \mathcal{A}) \tag{4.61}$$

where

$$\mathcal{Q}_1 \cap \mathcal{A} = \{ x = x_0 + C_{\mathcal{N}} z \mid \hat{z}^T M^T \hat{P}_1 M \hat{z} \leq 0 \} \tag{4.62}$$

$$\mathcal{Q}_2 \cap \mathcal{A} = \{ x = x_0 + C_{\mathcal{N}} z \mid \hat{z}^T M^T \hat{P}_2 M \hat{z} \leq 0 \} \tag{4.63}$$

and

$$\hat{z} = \begin{pmatrix} z \\ 1 \end{pmatrix} \quad , \quad M = \begin{pmatrix} C_{\mathcal{N}} & C^{\dagger} d \\ 0 & 1 \end{pmatrix} \tag{4.64}$$

Depending on the value of the power parameter r, we distinguish between four different cases.

- Case 1: $r \in (0, 1)$. Here, $\mathcal{Q}_2 = \emptyset$ and

$$\begin{gathered} \hat{P}_1 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & -1/2 \\ 0 & -1/2 & x_{n,0} \end{pmatrix}, \\ x_{n,0} = \frac{\overline{x}_n \underline{x}_n^r - \underline{x}_n \overline{x}_n^r}{\underline{x}_n^r - \overline{x}_n^r} \quad , \quad \alpha = \frac{\underline{x}_n - x_{n,0}}{\underline{x}_n^r} \end{gathered} \tag{4.65}$$

- Case 2: $r \in (1, 2)$. $\mathcal{Q}_2 = \emptyset$ and

$$\begin{gathered} \hat{P}_1 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} = \frac{\widetilde{x}_n \underline{x}_n^{r/2} - \underline{x}_n \widetilde{x}_n^{r/2}}{\underline{x}_n^{r/2} - \widetilde{x}_n^{r/2}} \quad , \quad \alpha = \frac{(\underline{x}_n - x_{n,0})^2}{\underline{x}_n^r} \end{gathered} \tag{4.66}$$

- Case 3: $r \in (2, \infty)$. Here we need to distinguish between three cases, depending on the number of positive real solutions to (4.78).
  - 1 solution: Here, $\mathcal{Q}_2 = \emptyset$ and

$$\begin{aligned} \hat{P}_1 &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}}. \end{aligned} \tag{4.67}$$

  - $\tilde{C}P^{-1}\tilde{C}^T = 0$: $\mathcal{Q}_2 = \emptyset$ and

$$\hat{P}_1 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/d^r \tag{4.68}$$

  - 3 solutions: Here,

$$\begin{aligned} \hat{P}_1 &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{\overline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \overline{x}_n^{r-2}} \end{aligned} \tag{4.69}$$

    and

$$\hat{P}_2 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/\underline{x}_n^r \tag{4.70}$$

- Case 4: $r \in (-\infty, 0)$. Here we have a representation analog to that for case 3 except for some minor changes. That is, we distinguish between three cases,
  - 1 solution: Here, $\mathcal{Q}_2 = \emptyset$ and

$$\begin{aligned} \hat{P}_1 &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}}. \end{aligned} \tag{4.71}$$

  - $\tilde{C}P^{-1}\tilde{C}^T = 0$: $\mathcal{Q}_2 = \emptyset$ and

$$\hat{P}_1 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/d^r \tag{4.72}$$

– 3 solutions: Here,

$$\hat{P}_1 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \qquad x_{n,0} = \frac{\underline{x}_n(r-2)}{r}, \quad \alpha = \frac{4}{r^2 \underline{x}_n^{r-2}} \tag{4.73}$$

and

$$\hat{P}_2 = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/\overline{x}_n^r \tag{4.74}$$

Using the value of r to distinguish between the four cases discussed in the two previous sections enables us to represent the intersection between a power cone and an affine set in an approximative manner as an affine or quadratic set, or the union of both. All approaches rely on finding certain values of x_n such that the intersection is known to be either below or above the hyperplane perpendicular to the x_n axis and intersecting it at that value. However, we have yet to specify how these values are to be calculated. This is the topic of the next section.

4.2.3 Computing Critical Values of x_n

The computations involved in finding outer and inner approximations of the intersection $\mathcal{PC} \cap \mathcal{A}$ utilize that values of x_n can be found such that components of $\mathcal{PC} \cap \mathcal{A}$ are located where x_n is either smaller or larger than the specific values. This section explains how these values can be obtained.

Lemma 4.1
Consider the intersection $\mathcal{PC} \cap \mathcal{A}$ where

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r \ , \ x_n \geq 0\} \tag{4.75}$$

with $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and x_n, $r \in \mathbb{R}$, and

$$\mathcal{A} = \{x \in \mathbb{R}^n \mid Cx = d\} \tag{4.76}$$

where $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^n$.

Let $\mathcal{H}_{x_n}$ denote a hyperplane perpendicular to the x_n axis and intersecting it at a constant value of x_n. The upper and lower bounds of x_n such that

$$\mathcal{H}_{x_n} \cap (\mathcal{PC} \cap \mathcal{A}) \neq \emptyset \tag{4.77}$$

are given by the positive real solutions to the equation

$$x_n^r = \beta x_n^2 + \eta x_n + \delta \tag{4.78}$$

where

$$\begin{aligned} \beta &= C_n^T(\tilde{C}P^{-1}\tilde{C}^T)^{-1}C_n \\ \eta &= -2d^T(\tilde{C}P^{-1}\tilde{C}^T)^{-1}C_n \\ \delta &= d^T(\tilde{C}P^{-1}\tilde{C}^T)^{-1}d \end{aligned} \tag{4.79}$$

and we have written $C = (\tilde{C}, C_n)$ *with* $\tilde{C} \in \mathbb{R}^{m \times n-1}$ *and* $C_n \in \mathbb{R}^m$.

Proof Let $\hat{x} = (\tilde{x}, x_n)$ be a point in $\mathcal{PC} \cap \mathcal{A}$ such that x_n satisfies (4.77). Obviously, $\hat{x} \in \partial(\mathcal{PC} \cap \mathcal{A})$. Hence, we seek the largest and smallest values of x_n such that $\hat{x}$ is on the boundary of $\mathcal{PC} \cap \mathcal{A}$. That is, we want to solve the problem

$$\begin{array}{ll} \text{max (min)} & x_n \\ \text{subject to} & \tilde{x}^T P \tilde{x} = x_n^r, \\ & Cx = d. \end{array} \tag{4.80}$$

Since this is an optimization problem with equality constraints, we can solve it using Lagrange multipliers. We form the Lagrangian

$$L(x, \lambda_1, \lambda_2) = x_n + \lambda_1(\tilde{x}^T P \tilde{x} - x_n^r) + \lambda_2^T(\tilde{C}\tilde{x} + C_n x_n - d) \tag{4.81}$$

where $\lambda_1 \in \mathbb{R}$, $\lambda_2 \in \mathbb{R}^m$. The necessary conditions for an extremum are

$$\frac{\partial L}{\partial \tilde{x}} = 0 = 2\lambda_1 P\tilde{x} + \tilde{C}^T\lambda_2 \tag{4.82}$$

$$\frac{\partial L}{\partial x_n} = 0 = 1 - r\lambda_1 x_n^{r-1} + C_n^T\lambda_2 \tag{4.83}$$

leading to

$$\tilde{x} = \frac{-P^{-1}\tilde{C}^T\lambda_2}{2\lambda_1} \tag{4.84}$$

$$x_n = \left(\frac{C_n^T\lambda_2 + 1}{r\lambda_1}\right)^{\frac{1}{r-1}}, \tag{4.85}$$

and

$$\frac{\partial L}{\partial \lambda_1} = 0 = \tilde{x}^T P \tilde{x} - x_n^r = \frac{\lambda_2^T \tilde{C}P^{-1}\tilde{C}^T\lambda_2}{4\lambda_1^2} - x_n^r \tag{4.86}$$

$$\frac{\partial L}{\partial \lambda_2} = 0 = \tilde{C}\tilde{x} + C_n x_n - d = \frac{-\tilde{C}P^{-1}\tilde{C}^T\lambda_2}{2\lambda_1} + C_n x_n - d. \tag{4.87}$$

Eliminating λ_2/λ_1 finally yields equation (4.78). □

Equation (4.78) is a nonlinear equation in one variable and can easily be solved numerically. Furthermore, the characteristics of (4.78) indicate whether $\mathcal{PC} \cap \mathcal{A}$ consists of one ore two connected components in cases 3 and 4:

- Equation (4.78) has one positive real solution: $\mathcal{PC} \cap \mathcal{A}$ consists of one unbounded component,

- $\tilde{C}P^{-1}\tilde{C}^T = 0$: This indicates that $\mathcal{A}$ is perpendicular to the x_n axis and hence $\mathcal{PC} \cap \mathcal{A}$ consists of one quadratic bounded component or the origin,

- Equation (4.78) has three positive real solutions: $\mathcal{PC} \cap \mathcal{A}$ consists of two unconnected components, one bounded and one unbounded.

If there is no solution, the power cone does not intersect the affine set.

The above catergorization implies that *the choice of approximation method can be automated* simply by examining equation (4.78) and selecting the appropriate method. Let us look at an example where this is applied to the power cone of Example 4.2.

Example 4.3 *Consider again the power cone and hyperplanes of Example 4.2. Starting with the hyperplane yielding one component, we solve (4.78) and find that* $\underline{x}_n = 1.7538$ *is the only positive real solution. Hence, we know that the intersection will lie in the halfspace defined by* $x_n \geq 1.7538$. *In the case of two components, (4.78) yields three positive solutions,* $\{\underline{x}_n, \widetilde{x}_n, \overline{x}_n\} = \{0.4319, 0.6582, 2.0804\}$. *Therefore, one component will lie in the halfspace defined by* $x_n \geq 2.0804$ *and the other in the cone*

$$\mathcal{Q} = \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \mid \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T 1.0901 \begin{pmatrix} 2 & 1/2 \\ 1/2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq (x_3 - 0.2608)^2 \}. \quad (4.88)$$

The case of a hyperplane perpendicular to the x_n *axis is identified by checking* c_1 *and we obtain the exact description*

$$\mathcal{PC} \cap \mathcal{A} = \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \mid \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 2 & 1/2 \\ 1/2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq 0.625 \}. \quad (4.89)$$

The categorization we have made here in terms of easily computed quantities enables us to algorithmically compute approximative representations of sets of the form $\mathcal{PC} \cap \mathcal{A}$. These representations can be expressed as affine and/or quadratic inequalities and hence allow us to do further computations using efficient tools. The reason why this type of sets is important will become apparent in Chapter 10 where we describe a procedure for automated verification.

4.3 Set Inclusion

We will need to be able to check if some specific sets are subsets of a power cone, here we describe methods that provide sufficient conditions for that in the case of polytopes and the intersection of a quadratic set with an affine set.

4.3.1 Polytope in Power Cone

Given a polytope as the convex hull of points, $\mathcal{P} = \mathbf{Co}\{v_1, \dots, v_p\}$, and a convex power cone, i.e.,

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \le x_n^r,\ x_n \ge 0\} \tag{4.90}$$

where $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$, $x_n \in \mathbb{R}$ and $r \in [0, 2]$, we know that $\mathcal{P} \subseteq \mathcal{PC}$ if and only if

$$\tilde{v}_i^T P \tilde{v}_i \le v_{n,i}^r,\ \forall\, i \in \{1, \dots, p\} \tag{4.91}$$

In case $r \notin [0, 2]$, the resulting power cone is non-convex and hence the above method will not work. Instead, we construct a quadratic cone $\mathcal{Q}$ which is a subset of $\mathcal{PC}$ and has the same cross section as $\mathcal{PC}$ at a specific x_n coordinate, $x_n = a$, see Figure 4.7. Hence, $\mathcal{P} \subseteq \mathcal{Q}$ implies that $\mathcal{P} \subseteq \mathcal{PC}$.

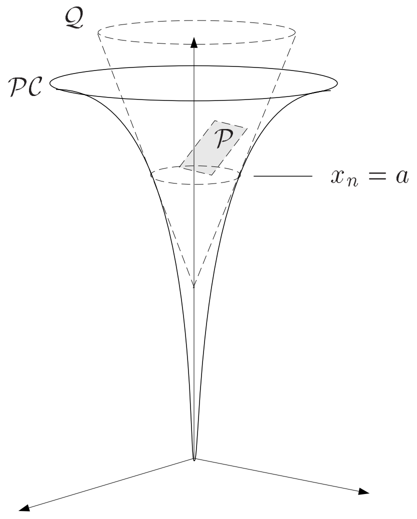


Figure 4.7: A quadratic cone contained in a non-convex power cone can be used to check if a polytope is a subset of the power cone.

We thus need conditions that enable us to determine if a quadratic cone is contained in a non-convex power cone while having a specific ellipsoid $\mathcal{E}_a$ as its cross section.

Lemma 4.2
Consider the power cone

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \le x_n^r,\ x_n \ge 0\} \tag{4.92}$$

where $r \in (2, \infty)$ (or $r \in (-\infty, 0)$). Let

$$\mathcal{H}_a = \{(\tilde{x}, x_n)^T \in \mathbb{R}^n \mid x_n = a\} \tag{4.93}$$

and define the ellipsoid $\mathcal{E}_a = \mathcal{PC} \cap \mathcal{H}_a$. Furthermore, let

$$\mathcal{Q} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T(\alpha P)\tilde{x} \leq (x_n - x_{n,0})^2\} \tag{4.94}$$

with $x_n \geq x_{n,0}$ ($x_n \leq x_{n,0}$), be a quadratic cone where

$$x_{n,0} = \frac{a(r-2)}{r}, \quad \alpha = \frac{(a - x_{n,0})^2}{a^r} = \frac{4}{r^2 a^{r-2}}. \tag{4.95}$$

Then $\mathcal{Q} \subseteq \mathcal{PC}$ and $\mathcal{Q} \cap \mathcal{H}_a = \mathcal{E}_a$.

Proof That $\mathcal{Q} \cap \mathcal{H}_a = \mathcal{E}_a$ follows directly from inserting $x_n = a$ into the indicator functions of $\mathcal{PC}$ and $\mathcal{Q}$. This gives

$$\tilde{x}^T P \tilde{x} = a^r = \frac{(a - x_{n,0})^2}{\alpha} \tag{4.96}$$

which leads to

$$\alpha = \frac{(a - x_{n,0})^2}{a^r}. \tag{4.97}$$

We then apply the S-procedure of Section 3.3.1 to show that

$$\tilde{x}^T P \tilde{x} - x_n^r \leq 0 \quad \forall x \text{ s.t. } \tilde{x}^T(\alpha P)\tilde{x} - (x_n - x_{n,0})^2 \leq 0. \tag{4.98}$$

Hence, we require that there exists a $\tau \geq 0$ such that

$$\tilde{x}^T P \tilde{x} - x_n^r - \tau(\tilde{x}^T(\alpha P)\tilde{x} - (x_n - x_{n,0})^2) \leq 0. \tag{4.99}$$

Taking $\tau = 1/\alpha$ and inserting the expressions for $x_{n,0}$ and α, we find after straightforward manipulations that (4.99) is implied by

$$x_n^{r/2} \geq \frac{1}{2} a^{r/2-1} r x_n - \frac{1}{2} a^{r/2}(r-2). \tag{4.100}$$

That is, the power function should be greater than a linear function for all values of $a, x_n \geq 0$ and $r \notin [0, 2]$ for the subset relation to hold. Since the former is monotonically increasing (decreasing) and has a monotonically increasing derivative for all $r > 2$, ($r < 0$), it suffices to show that there exists exactly one point where the linear function is tangent to the power function. Equating the derivatives of the two functions, we find that they have the same slope only at $x_n = a$ which inserted into (4.100) yields equality. $\square$

All that remains is to make sure that we select the value a such that the ellipsoid $\mathcal{E}_a$ has minimal volume amongst those that intersect the polytope $\mathcal{P}$. This is expressed in the following corollary.

Corollary 4.1
Consider the polytope

$$\mathcal{P} = \mathbf{Co}\{v_1, \dots, v_p\} \tag{4.101}$$

and the power cone

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r\} \tag{4.102}$$

If $r \in [0, 2]$, then $\mathcal{P} \subseteq \mathcal{PC}$ if and only if

$$\tilde{v}_i^T P \tilde{v}_i \leq x_{n,i}^r, \ \forall\, i \in \{1, \dots, p\}. \tag{4.103}$$

Otherwise, if $r \notin [0, 2]$, $\mathcal{P} \subseteq \mathcal{PC}$ provided that

$$\tilde{v}_i^T (\alpha P) \tilde{v}_i \leq (v_{n,i} - x_{n,0})^r, \ \forall\, i \in \{1, \dots, p\} \tag{4.104}$$

where

$$x_{n,0} = \frac{a(r-2)}{r}, \quad \alpha = \frac{4}{r^2 a^{r-2}}. \tag{4.105}$$

and

$$a = \min_{i \in \{1, \dots, p\}} (v_i)_n \tag{4.106}$$

if $r \in (2, \infty)$ and

$$a = \max_{i \in \{1, \dots, p\}} (v_i)_n \tag{4.107}$$

if $r \in (-\infty, 0)$.

Let us now take a look at how we can check if the intersection of a quadratic set and an affine set is contained in a power cone.

4.3.2 Quadratic and Affine Set in Power Cone

We need to be able to check if a quadratic set intersected with an affine set is contained in a power cone, i.e., if $\mathcal{Q} \cap \mathcal{A} \subseteq \mathcal{PC}$ where

$$\begin{aligned} \mathcal{Q} &= \{x \in \mathbb{R}^n \mid x^T A x + 2b^T x + c \leq 0\} \\ \mathcal{A} &= \{x \in \mathbb{R}^n \mid Cx = d\} \end{aligned} \tag{4.108}$$

with $A = A^T \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$, $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^m$ where m is the number of inequalities defining the affine set. Furthermore,

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r, \ x_n \geq 0\} \tag{4.109}$$

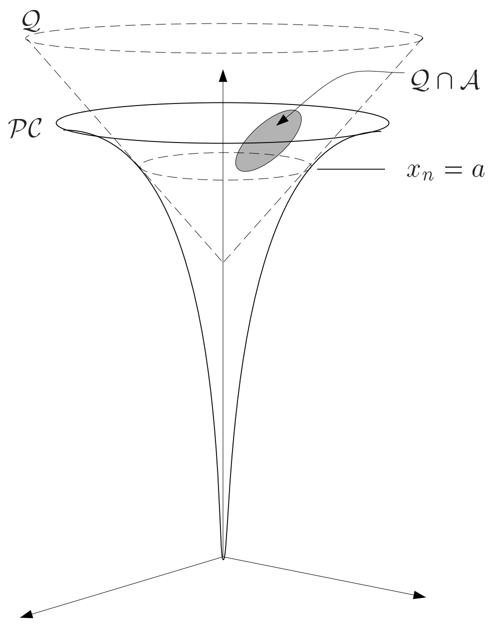


Figure 4.8: An example of a quadratic and affine set contained in a power cone when $r \in (2, \infty)$.

where $P \in \mathbb{R}^{n-1 \times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and $x_n,\ r \in \mathbb{R}$.

Let us start by considering Figure 4.8 where we see an example of a quadratic and affine set contained in a power cone. The basic idea is as follows. First we find the value $x_n = a$ where the ellipsoid $\mathcal{E}_a^{\mathcal{PC}}$, obtained by intersecting $\mathcal{PC}$ with the hyperplane $\mathcal{H}_a = \{x \in \mathbb{R}^n \mid x_n = a\}$, is smallest while $\mathcal{H}_a \cap (\mathcal{Q} \cap \mathcal{A}) \neq \emptyset$. We then find the largest quadratic set that has $\mathcal{E}_a^{\mathcal{PC}}$ as its cross section and is guaranteed to be a subset of $\mathcal{PC}$. Which type of quadratic set we use depends on the value of the power parameter r, we select a cylinder if $r \in (0, 1)$, a paraboloid if $r \in (1, 2)$ and a cone if $r \notin [0, 2]$. Finally, we check if this quadratic set contains $\mathcal{Q} \cap \mathcal{A}$ and if this is the case we know that so does the power cone.

Let us start by looking at the different cases that arise when finding the largest quadratic set contained in the power cone.

The case when $r \in (0, 1)$ is the simplest one, we simply use the cylinder having the ellipsoid $\mathcal{E}_a^{\mathcal{PC}}$ as its cross section, i.e.,

$$\mathcal{Q} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T \tilde{P} \tilde{x} \leq a^r\}. \tag{4.110}$$

When $r \notin [0, 2]$, we can use Lemma 4.2 to construct an elliptic cone that has $\mathcal{E}_a^{\mathcal{PC}}$ as its cross section and is contained in $\mathcal{PC}$.

For the case when $r \in (1,2)$, we need the equivalent of Lemma 4.2 but formulated for paraboloids instead of cones.

Lemma 4.3
Consider the power cone

$$\mathcal{PC} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r,\ x_n \geq 0\} \tag{4.111}$$

where $r \in (1,2)$. *Let*

$$\mathcal{H}_a = \{(\tilde{x}, x_n)^T \in \mathbb{R}^n \mid x_n = a\} \tag{4.112}$$

and define the ellipsoid $\mathcal{E}_a = \mathcal{PC} \cap \mathcal{H}_a$. *Furthermore, let*

$$\mathcal{Q} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T (\alpha P) \tilde{x} \leq x_n - x_{n,0}\} \tag{4.113}$$

with $x_n \geq x_{n,0}$, *be a paraboloid where*

$$x_{n,0} = \frac{a(r-1)}{r}, \quad \alpha = \frac{a - x_{n,0}}{a^r} = \frac{1}{ra^{r-1}}. \tag{4.114}$$

Then $\mathcal{Q} \subseteq \mathcal{PC}$ *and* $\mathcal{Q} \cap \mathcal{H}_a = \mathcal{E}_a$.

Proof That $\mathcal{Q} \cap \mathcal{H}_a = \mathcal{E}_a$ follows directly from inserting $x_n = a$ into the indicator functions of $\mathcal{PC}$ and $\mathcal{Q}$. This gives

$$\tilde{x}^T P \tilde{x} = a^r = \frac{a - x_{n,0}}{\alpha} \tag{4.115}$$

which leads to

$$\alpha = \frac{a - x_{n,0}}{a^r}. \tag{4.116}$$

We then apply the S-procedure of Section 3.3.1 to show that

$$\tilde{x}^T P \tilde{x} - x_n^r \leq 0 \quad \forall x \text{ s.t. } \tilde{x}^T (\alpha P) \tilde{x} - x_n + x_{n,0} \leq 0. \tag{4.117}$$

Hence, we require that there exists a $\tau \geq 0$ such that

$$\tilde{x}^T P \tilde{x} - x_n^r - \tau(\tilde{x}^T (\alpha P) \tilde{x} - x_n + x_{n,0}) \leq 0. \tag{4.118}$$

Taking $\tau = 1/\alpha$ and inserting the expressions for $x_{n,0}$ and α, we find after straightforward manipulations that (4.118) is implied by

$$x_n^r \geq a^{r-1} r x_n - a^r (r-1). \tag{4.119}$$

That is, the power function should be greater than a linear function for all values of $a, x_n \geq 0$ and $r \in (1,2)$ for the subset relation to hold. Since the former is monotonically increasing and has a monotonically increasing derivative for all $r \in (1,2)$, it suffices to show that there exists exactly one point where the linear function is tangent to the power function. Equating the derivatives of the two functions, we find that they have the same slope only at $x_n = a$ which inserted into (4.119) yields equality. $\square$

We thus have the means of using the power parameter r to find a suitable quadratic set contained in $\mathcal{PC}$ once we know the x_n value resulting in the smallest ellipsoidal cross section. We now need a way to determine that value.

The value of x_n where the ellipsoid $\mathcal{E}_{x_n}^{\mathcal{PC}}$ is smallest can be found by solving an optimization problem with equality constraints. This is stated in the following lemma.

Lemma 4.4
Consider the intersection $\mathcal{Q} \cap \mathcal{A}$ where

$$\begin{aligned}\mathcal{Q} &= \{x \in \mathbb{R}^n \mid x^T A x + 2b^T x + c \leq 0\} \\ &= \{\begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix} \in \mathbb{R}^n \mid \begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix}^T \begin{pmatrix}\tilde{A} & \tilde{a} \\ \tilde{a}^T & a_{nn}\end{pmatrix} \begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix} + 2\begin{pmatrix}\tilde{b} \\ b_n\end{pmatrix}^T \begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix} + c \leq 0\}\end{aligned} \tag{4.120}$$

with $\tilde{A} \in \mathbb{R}^{n-1 \times n-1}$, $\tilde{A} = \tilde{A}^T \succ 0$, $\tilde{a}, \tilde{x} \in \mathbb{R}^{n-1}$, $a_{nn}, b_n, c, x_n \in \mathbb{R}$ and

$$\begin{aligned}\mathcal{A} &= \{x \in \mathbb{R}^n \mid Cx = d\} \\ &= \{\begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix} \in \mathbb{R}^n \mid (\tilde{C}, C_n)\begin{pmatrix}\tilde{x} \\ x_n\end{pmatrix} = d\}\end{aligned} \tag{4.121}$$

where $\tilde{C} \in \mathbb{R}^{m \times n-1}$ and $C_n \in \mathbb{R}^m$. Let $\mathcal{H}_{x_n}$ denote a hyperplane perpendicular to the x_n axis and intersecting it at a constant value of x_n. The upper and lower bounds of x_n such that

$$\mathcal{H}_{x_n} \cap (\mathcal{Q} \cap \mathcal{A}) \neq \emptyset \tag{4.122}$$

are given by the real solutions to the second order equation

$$\alpha x_n^2 + \beta x_n + \gamma = 0 \tag{4.123}$$

where

$$\begin{aligned}\alpha &= a_n n - \tilde{a}^T \tilde{A}^{-1} \tilde{a} + (C_n - C_{\tilde{a}})^T C_{\tilde{A}} (C_n - C_{\tilde{a}}) \\ \beta &= 2(b_n - \tilde{b}^T \tilde{A}^{-1} \tilde{a} - d^T C_{\tilde{A}} (C_n - C_{\tilde{a}}) - C_{\tilde{b}}^T C_{\tilde{A}} (C_n - C_{\tilde{a}})) \\ \gamma &= c + d^T C_{\tilde{A}} d + C_{\tilde{b}}^T C_{\tilde{A}} C_{\tilde{b}} + 2d^T C_{\tilde{A}} C_{\tilde{b}}\end{aligned} \tag{4.124}$$

and

$$C_{\tilde{a}} = \tilde{C}\tilde{A}^{-1}\tilde{a}, \quad C_{\tilde{b}} = \tilde{C}\tilde{A}^{-1}\tilde{b}, \quad C_{\tilde{A}} = (\tilde{C}\tilde{A}^{-1}\tilde{C}^T)^{-1}. \tag{4.125}$$

Proof Let $\hat{x} = (\tilde{x}, x_n)$ be a point in $\mathcal{Q} \cap \mathcal{A}$ such that x_n satisfies (4.122). Obviously, $\hat{x} \in \partial(\mathcal{Q} \cap \mathcal{A})$. Hence, we seek the largest and smallest values of x_n such that $\hat{x}$ is on the boundary of $\mathcal{Q} \cap \mathcal{A}$. That is, we want to solve the problem

$$\begin{array}{ll}\max\ (\min) & x_n \\ \text{subject to} & a_{nn}x_n^2 + 2(b_n + \tilde{a}^T\tilde{x})x_n + \tilde{x}^T\tilde{A}\tilde{x} + 2\tilde{b}^T\tilde{x} + c = 0, \\ & \tilde{C}\tilde{x} + C_n x_n = d.\end{array} \tag{4.126}$$

Since this is an optimization problem with equality constraints, we can solve it using Lagrange multipliers. We form the Lagrangian

$$L(x,\lambda_1,\lambda_2) = x_n + \lambda_1(a_{nn}x_n^2 + 2(b_n + \tilde{a}^T\tilde{x})x_n + \tilde{x}^T\tilde{A}\tilde{x} + 2\tilde{b}^T\tilde{x} + c) + \\ + \lambda_2^T(\tilde{C}\tilde{x} + C_n x_n - d) \tag{4.127}$$

where $\lambda_1 \in \mathbb{R}$, $\lambda_2 \in \mathbb{R}^m$ and pose the necessary conditions for an extremum

$$\frac{\partial L}{\partial \tilde{x}} = 0 = 2\lambda_1\tilde{a}x_n + 2\lambda_1\tilde{A}\tilde{x} + 2\lambda_1\tilde{b} + \tilde{C}^T\lambda_2 \tag{4.128}$$

$$\frac{\partial L}{\partial x_n} = 0 = 1 + 2\lambda_1 a_{nn}x_n + 2\lambda_1(b_n + \tilde{a}^T\tilde{x}) + C_n^T\lambda_2 \tag{4.129}$$

$$\frac{\partial L}{\partial \lambda_1} = 0 = a_{nn}x_n^2 + 2(b_n + \tilde{a}^T\tilde{x})x_n + \tilde{x}^T\tilde{A}\tilde{x} + 2\tilde{b}^T\tilde{x} + c \tag{4.130}$$

$$\frac{\partial L}{\partial \lambda_2} = 0 = \tilde{C}\tilde{x} + C_n x_n - d. \tag{4.131}$$

Solving for $\tilde{x}$ from (4.128) gives

$$\tilde{x} = -\tilde{A}^{-1}(\tilde{a}x_n + \tilde{b} + \frac{\tilde{C}^T\lambda_2}{2\lambda_1}) \tag{4.132}$$

which inserted into (4.130) and (4.131) results in

$$(a_{nn} - \tilde{a}^T\tilde{A}^{-1}\tilde{a})x_n^2 + 2(b_n - \tilde{b}^T\tilde{A}^{-1}\tilde{a})x_n + c + \frac{\lambda_2^T\tilde{C}\tilde{A}^{-1}\tilde{C}^T\lambda_2}{4\lambda_1^2} = 0 \tag{4.133}$$

and

$$\frac{\tilde{C}\tilde{A}^{-1}\tilde{C}^T\lambda_2}{2\lambda_1} = (C_n - \tilde{C}\tilde{A}^{-1}\tilde{a})x_n - d - \tilde{C}\tilde{A}^{-1}\tilde{b}. \tag{4.134}$$

These two equations enable us to eliminate λ_1 and λ_2 which after some tedious but straightforward manipulations finally yields equation (4.123). □

We can use Lemma 4.4 to find the extreme values of x_n, we then select the one that results in the smallest ellipsoid $\mathcal{E}_{x_n}^{\mathcal{PC}}$, that is the larger value if $r \in (-\infty, 0)$ and the smaller value if $r \in (0, \infty)$.

Note that Lemma 4.4 is limited to quadratic sets where $\tilde{A}$ is positive definite. This does not pose a problem since that condition will be fulfilled for those sets to which we intend to apply the lemma.

The set inclusion can now be determined by checking if there exists a value $\tau \in \mathbb{R}$ such that

$$M^T\left(\hat{P} - \tau\begin{pmatrix} A & b \\ b^T & c \end{pmatrix}\right)M \preceq 0, \quad \tau \geq 0 \tag{4.135}$$

where

$$M = \begin{pmatrix} C_{\mathcal{N}} & C^{\dagger}d \\ 0 & 1 \end{pmatrix} \tag{4.136}$$

and $\hat{P}$ varies depending on the value of the power parameter r according to the following:

- $r \in (0,1)$:

$$\hat{P} = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/a^r \tag{4.137}$$

- $r \in (1,2)$:

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix} , \\ x_{n,0} &= \frac{a(r-1)}{r} \quad , \quad \alpha = \frac{1}{ra^{r-1}} \end{aligned} \tag{4.138}$$

- $r \notin [0,2]$:

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & -1/2 \\ 0 & -1/2 & x_{n,0} \end{pmatrix} , \\ x_{n,0} &= \frac{a(r-2)}{r} \quad , \quad \alpha = \frac{4}{r^2 a^{r-2}} \end{aligned} \tag{4.139}$$

where a is the larger value of the solutions to equation (4.123) if $r \in (-\infty, 0)$ and the smaller value if $r \in (0, \infty)$.

4.4 Summary

In this chapter we have described how power cones and their intersection with affine sets can be represented. The latter are approximated using quadratic and affine sets where different sets are appropriate in different situations. We have developed methods for distinguishing algorithmically between these situations based on simple analytical expressions and/or numerical solutions to equations in one variable. Hence, automatic construction of both outer and inner approximations is made possible. Sufficient conditions for the inclusion of polytopes and the intersection of quadratic and affine sets in power cones are also developed. So are systematic methods for automatically selecting appropriate conditions. These topics serve as computational tools for the model checking methods described in Chapter 10.

Part II

Modeling

5 Modeling of Dynamic Systems

The intention with this chapter is to give some background to topics related to modeling of dynamic systems that we utilize in this thesis, as well as introducing a somewhat novel framework for a specific class of dynamic systems. Since we are treating a topic created by the overlap of two disciplines, we run the risk of elaborating on items of elementary nature in, at least, one of the disciplines. We take that risk, but limit our introduction to material which is essential to the topics treated in subsequent chapters.

In Section 5.1 we give an introduction to some different formalisms for modeling of dynamic systems in both discrete and continuous time, and in Section 5.2 we introduce a specific framework for modeling piecewise linear systems.

5.1 Dynamic Systems

Our mental picture of a system is as an object of some sort where we have some interaction of variables. We commonly divide these variables into inputs stimulating the system, and outputs which constitute our observations of the system behavior. If this behavior depends solely on the input at the current time instance, we call the system static, otherwise, if the behavior is also influenced by inputs applied at previous time instances, we call the system dynamic. In order to eliminate the need for keeping track of previous inputs to a dynamic system, we often find it convenient to introduce variables having the property that they contain all relevant information about the effect of previous inputs on the system. These variables are called state variables since they contain all we need to know about the "state" of the system in order to be able to determine future behavior for a specified input.

We can describe dynamic systems using models capturing relevant behavior at different levels of abstraction. The choice of description (model) should of course reflect the intended use and naturally we get a wide spectrum of models depending on context.

For instance, we may have conceived a mental model of a non-existing technical system. This model may be communicated using different kinds of informal descriptions and as a more or less detailed technical specification as we move towards realization of the system. Reaching the design stages, we are likely to produce, again more or less detailed, mathematical descriptions which serve as aid during the design process. It is such mathematical models we usually refer to when speaking of models and we will in the following sections explore different kinds of mathematical models as means for specifying, analyzing and designing dynamic systems.

Remark 5.1
In the sequel, we will frequently use the word system when we actually mean a model of a system. In cases where the distinction is important, we will clarify what we mean.

5.1.1 Discrete Event Systems

At the upper end of the abstraction scale, we find models which describe dynamic systems as objects having discrete inputs and outputs. Furthermore, such a system can be in one of a finite number of internal configurations, or states, and as events arrive at discrete (sporadic) time instances at the input, the system may change its state and produce an output. We refer to systems suitable for such a description as Discrete Event Dynamic System, or simply DEDS.

A formal definition of DEDS in line with that of [52] is given below.

Definition 5.1 (DEDS)
A Discrete Event Dynamic System is a dynamic system that evolves from state to state in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of input events, producing output events.

The state of a DEDS is represented (encoded) by the state variable $x \in \mathcal{X}$ where $\mathcal{X}$ is the state domain. The system accepts input events, represented by the input variables $u \in \mathcal{U}$ and produces output events, represented by the output variables $y \in \mathcal{Y}$, where $\mathcal{U}$ and $\mathcal{Y}$ are the input and output domains, respectively.

Different formalisms exist for describing DEDS, each imposing a tradeoff between modeling capability and analysis or design methods. Examples are Petri nets [53] and its relative Grafcet [3], Statecharts [25] and, our choice, finite state machines [21], [31].

Definition 5.2 (Finite State Machine)
A finite state machine (FSM) $\mathcal{M}$ is a collection of six objects, $\mathcal{M} = \{\mathcal{X}, x^0, \mathcal{U}, f, \mathcal{Y}, h\}$ where

- *$\mathcal{X}$ is a set of states,*
- *x^0 is an initial state,*
- *$\mathcal{U}$ is a set of inputs,*
- *$f : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}}$ is a transition relation defining how the next state x^+ depends on the current state x and input u,*
- *$\mathcal{Y}$ is a set of outputs, and*
- *$h : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ is an output function associating an output y to each pair (x, u) of current state and input.*

When the output function, h, of a finite state machine is limited to a mapping from states only to output, i.e., $h : \mathcal{X} \to \mathcal{Y}$, the FSM is called a Moore machine, otherwise, it is called a Mealy machine. Although different, the two types of machines can be seen as equivalent in the sense that they produce the same input-output mapping.

It is common practice to visualize a FSM by associating with it a directed graph. This graph is called a transition diagram and is constructed as follows. The vertices correspond to the states of the FSM and each vertex is labeled to indicate the corresponding state. In addition, the initial state is distinguished by an arrow pointing at the corresponding vertex. When a transition exists from a state x^1 to a state x^2 on input u^1, there is an arc labeled u^1 from x^1 to x^2. Furthermore, we place a label indicating the output produced by the state machine. In the case of Moore machines, the label is put at a vertex together with the state label, that is we write x^i/y^i at the vertex. In the case of Mealy machines, the label is placed together with the input at the arc representing the transition for which the output is defined. That is, we write u^i/y^i at the corresponding arc.

The above definition and discussion is best illustrated with an example.

Example 5.1 (Transition Diagrams) *Let us consider finite state machines with states $\mathcal{X} = \{a, b, c\}$ where $x^0 = a$ is an initial state. Define the input alphabet as $\mathcal{U} = \{0, 1\}$ and the output alphabet $\mathcal{Y} = \{r, s\}$. A FSM of Mealy type is shown in Figure 5.1 while a finite state machine of Moore type is shown in Figure 5.2.*

Remark 5.2
Note that the two state machines shown in the example are by no means equivalent, the intention is merely to clarify the syntax used for different types of FSMs.

A special case of a finite state (Moore) machine is the finite automaton [35]. Its output is limited to a binary symbol, “accept”/“don’t accept” depending on whether the automaton has reached one of specially labeled final states, $\mathcal{X}^1$. Hence, we can

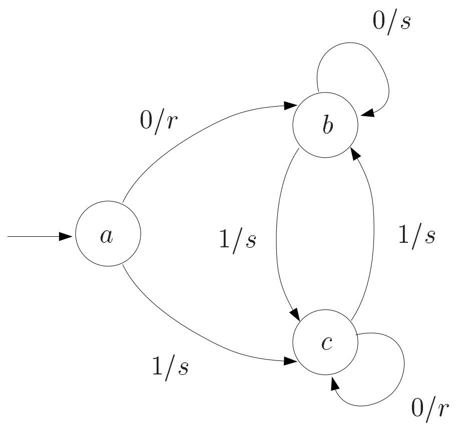


Figure 5.1: A Mealy machine.

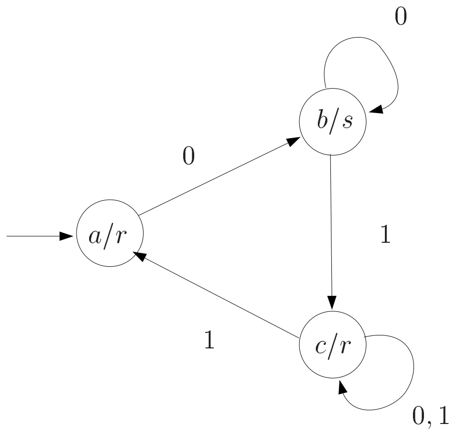


Figure 5.2: A Moore machine.

derive the finite automaton as a Moore machine where we have two possible outputs, $\mathcal{Y} = \mathbb{B} = \{\texttt{true}, \texttt{false}\}$, and define the output function as $h(x) = X^1(x)$, resulting in the value `true` when a state fulfills the condition on final states [31].

Example 5.2 (A Finite Automaton) *Consider a finite automaton over the same sets $\mathcal{X}$,$\mathcal{U}$ as the state machines in Example 5.1 and with the same initial state x^0. Replace the set of outputs by $\mathcal{Y} = \mathbb{B}$. Define c as a final state, i.e., the set of final states as $\mathcal{X}^1 = \{x \in \mathcal{X} \mid X^1(x)\} = \{x \in \mathcal{X} \mid [x = c]\}$. We now have the transition diagram shown in Figure 5.3 where the final state is distinguished using* *double circles. We now see that in order to use this device for detecting when we*

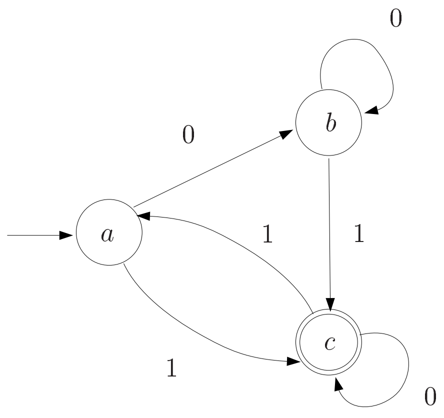


Figure 5.3: A finite automaton.

enter the final state, we can define the output function as $h(x) = X(x) = [x = c]$ yielding the output $y = \texttt{true}$ if and only if the current state is c.

Remark 5.3
Definition 5.2 describes non-deterministic finite state machines. In the deterministic case, the transition relation f defines a unique next state for each pair of state and input, i.e., $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. For instance, the state machine in Figure 5.1 is deterministic while the FSM in Figure 5.2 is non-deterministic due to the transition from c on the input 1 which may either lead us to the state a or stay put in state c.

To illustrate how we can model engineering systems using finite state machines, we will model a simple water tank at this relatively high level of abstraction.

Example 5.3 (Water Tank - Discrete Model) *Consider the water tank and input valve in Figure 5.4. The tank can be full, empty or neither, we thus define the states of the DEDS as $\mathcal{X} = \{\texttt{empty}, \texttt{normal}, \texttt{full}\}$. This set can be encoded using two binary variables, $x_1, x_2 \in \mathbb{B}$ and a code function (see Section 2.2) $\lambda(x)$ in accordance with Table 5.1, yielding the representation $\mathcal{X} = \{\lambda(x) \mid x \in \mathbb{B}^2, \neg[x_1 \wedge x_2]\}$. The valve can be either open or closed. We model its position as an input to our system and let $u = \texttt{true}$ represent the open position and $u = \texttt{false}$ represent closed position, this of course implies that $\mathcal{U} = \{\texttt{true}, \texttt{false}\}$. We define our initial state $x^0 = (\texttt{false}, \texttt{false})^T$ corresponding to "normal" level, and the transition function according to Table 5.2. Finally, we may define the output function as $h(x) = x$, we could then apply the coding function λ to y for an interpretation of the output $y \in \mathcal{Y} = \mathbb{B}^2$.*

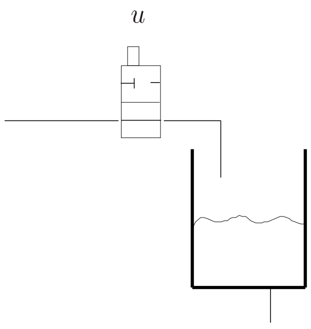


Figure 5.4: A water tank.

$\lambda(x)$		x_2 false	x_2 true
x_1	false	normal	full
	true	empty	-

Table 5.1: State encoding.

This simple example provides us with some insights into the effects of modeling at different levels of abstraction.

For instance, we find it quite natural to reason about the behavior of the system at the discrete level, mainly since concepts like "full tank" or "empty tank" have an intuitive meaning. Hence, we might suspect that it should be relatively easy to take the step from a specification of desired properties of the system, to a formal description at the discrete level. This is at least partially true, with the main problem being that the gap between an informal specification and a formal one may sometimes be quite large. However, we usually find that the discrete model conforms rather well with how we reason about logical behavior.

We also see that we need to sacrifice detailed information about the water level; there is no way of telling how the level behaves while in the "normal" state. Whether this is an issue or not depends of course on the context. If it is, we need to turn to a completely different modeling framework focusing on the continuous aspects of the system. This framework is discussed briefly in the next section.

5.1.2 Continuous Systems

Models of continuous time systems express in great detail the behavior of the system and can thus be seen as residing at the lower end of the abstraction scale. They are usually modeled using ordinary differential equations (ODEs). A common form of such a description for nonlinear time-invariant systems is the state space model.

$f(x,u)$		u	
		true	false
x	$\begin{pmatrix}\texttt{true}\\ \texttt{false}\end{pmatrix}$	$\begin{pmatrix}\texttt{false}\\ \texttt{false}\end{pmatrix}$	$\begin{pmatrix}\texttt{true}\\ \texttt{false}\end{pmatrix}$
	$\begin{pmatrix}\texttt{false}\\ \texttt{false}\end{pmatrix}$	$\begin{pmatrix}\texttt{false}\\ \texttt{true}\end{pmatrix}$	$\begin{pmatrix}\texttt{true}\\ \texttt{false}\end{pmatrix}$
	$\begin{pmatrix}\texttt{false}\\ \texttt{true}\end{pmatrix}$	$\begin{pmatrix}\texttt{false}\\ \texttt{true}\end{pmatrix}$	$\begin{pmatrix}\texttt{false}\\ \texttt{false}\end{pmatrix}$

Table 5.2: Tank transition function.

Definition 5.3 (State Space Model)
A state space model consists of two sets of equations, a set of first order continuous differential equations called state equations, and a set of output equations. That is,

$$\dot{x}(t) = f(x(t), u(t)) \tag{5.1}$$
$$y(t) = h(x(t), u(t)) \tag{5.2}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input vector and $y(t) \in \mathbb{R}^p$ is the output vector. Furthermore, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a vector field defining a transition map $\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ taking the state vector from an initial state x_0 at $t = t_0$ to the state $x(t)$ obtained as the solution to (5.1), and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ is an output function.

Nonlinear state space models often arise from a modeling procedure based on basic physical principles [9], [34], [41]. The construction is often aided using some modeling formalism, for instance, using physical Bond Graphs introduced in [46]. However, one can also apply system identification to estimate parameters of nonlinear "black box" structures [39], [40], or obtain the models from a combination of physical modeling and system identification [38].

When the vector field f and output function h are linear functions of the state and input vectors, we have a linear state space model

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned} \tag{5.3}$$

where $A \in \mathbb{R}^n \times \mathbb{R}^n$, $B \in \mathbb{R}^n \times \mathbb{R}^m$ and $C \in \mathbb{R}^p \times \mathbb{R}^n$. A linear model can be obtained, for instance, by a local approximation of a nonlinear one or directly from measured data using system identification to estimate parameters of linear model structures [39]. Comparing the linear and non-linear models, we find the same tradeoffs between modeling power and analysis/design methods as in the case of different formalisms for modeling discrete event systems. The theory of linear systems is an extremely well developed field and a multitude of methods exist for analysis and design [33], [54]. However, our main concern regarding their use is usually whether they are valid descriptions of our systems.

As an example of the advantages of working with linear systems, we mention that for the description (5.3), we directly have an expression for its solution, namely (with $D = 0$)

$$y(t) = Ce^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} Ce^{A(t-\tau)}Bu(\tau)d\tau \tag{5.4}$$

where e^{At} is the matrix exponential function.

Remark 5.4
Note that our definition of linear systems is the one from the control and systems theory community. That is, we refer to a system as linear if its output corresponding to an input of the form $c_1u_1(t) + c_2u_2(t)$ where c_1, c_2 are arbitrary constants, equals $c_1y_1(t) + c_2y_2(t)$ where $y_1(t), y_2(t)$ are the outputs for inputs $u_1(t)$ and $u_2(t)$, respectively. This is in contrast with the definition commonly used within the computer science community where a system is called linear if it generates trajectories which are linear functions of time [24], [28], [58]. Clearly, that definition corresponds merely to a special case of (5.4).

In case that we do not have any input to our system and the vector field is a linear function of the state vector, we obtain an uncontrolled linear state space model; a slightly more general model is obtained by allowing $f(x)$ to be affine in the state vector x.

Definition 5.4 (Affine State Space Model)
An affine state space model has the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + b \\ y(t) &= Cx(t) \end{aligned} \tag{5.5}$$

where $A \in \mathbb{R}^n \times \mathbb{R}^n$, $b \in \mathbb{R}^n$ and $C \in \mathbb{R}^p \times \mathbb{R}^n$.

This type of models may, for instance, be obtained from a linear model with (piecewise) constant inputs, $u(t) = \bar{u}$ for $t_1 \leq t < t_2$. In that case, the constant system matrices A, b are obtained as $e^{A(t_2-t_1)}$ and $\left(\int_{t_1}^{t_2} e^{A(t-\tau)}d\tau\right)B\bar{u}$, respectively. When $t_1, t_2, \ldots$ are equidistant sampling points and the input is constant between sampling points, the resulting description is a linear discrete time model and again we have a well established theory for dealing with such systems.

We conclude this section by exemplifying the use of differential equations for modeling engineering systems.

Example 5.4 (Water Tank, contd.) *Consider again the water tank process from example 5.3. Assuming that the pump flow Q_{in} can be varied continuously, and noting that the outflow Q_{out} is given by Bernoullis law, we arrive at the* *nonlinear first order differential equation describing the change in water level, h, as proportional to the difference between in- and outflow. That is,*

$$\dot{h} = \frac{1}{A}(Q_{in} - Q_{out}) = \frac{1}{A}(Q_{in} - a_d\sqrt{2gh}) \tag{5.6}$$

where A and a_d are the areas of the tank and outflow pipe, respectively, and g is the gravity constant. Now, this equation can be linearized around a stationary point $\dot{h} = 0$ for a given inflow $\bar{Q}_{in}$, this is the point $\bar{h} = \frac{\bar{Q}_{in}^2}{2ga_d^2}$. As a result, we get a first order linear differential equation describing variations around the stationary point, $x = h - \bar{h}$, $u = Q_{in} - \bar{Q}_{in}$ which we can write as

$$\dot{x} = ax + bu \tag{5.7}$$

where $a = -\frac{a_d\sqrt{2g}}{2A\sqrt{\bar{h}}}$ and $b = \frac{1}{A}$.

We see that we end up with a model giving a detailed description of the system where it is valid. Linear models are usually only valid locally in a certain operating region, but here even our nonlinear model is only valid while the tank does not become either full or empty. To cope with this aspect of discrete events occuring in an otherwise continuous framework, the notion of hybrid systems is introduced.

5.1.3 Hybrid Systems

The topic of hybrid systems deals with modeling, analysis and control of systems whose behavior is a combination of continuous evolution and abrupt changes.

Several special classes of hybrid systems have been treated recently, e.g., in [1] and [60] where systems with piecewise linear trajectories are considered, and in [2] where systems which can be modeled as discrete event systems with timing information (timed automata) are treated. In [32], methods for verification of piecewise affine systems are suggested. An overview of general hybrid systems can, for instance, be obtained in [6].

A special class of hybrid systems are switched systems, where system trajectories are continuous but may have discontinuous derivatives. We intend to focus on this class in this thesis, and in particular those systems where the dynamics are governed by piecewise linear differential equations. A framework for these systems is given in the next section.

5.2 A Framework for Switched Systems

Piecewise linear systems have been treated, for instance, in [48], [49] and [47], where graphical analysis methods are developed, and in [55] where several results are provided for the discrete time case. The modeling framework for piecewise linear systems introduced in this thesis originates from the framework for hybrid systems developed in [57]; it also contains elements inspired by [20].

The hybrid systems considered are viewed as composed of three main components: a continuous time plant, a discrete event controller, and an interface. The interface provides communication between the plant and the controller by converting signals from the continuous domain of the former to the discrete, symbolic domain of the latter, and vice versa. The interface can be further decomposed into an actuator translating controller symbols to plant input signals and a generator transforming continuous signals to discrete symbols used by the controller.

The modeling framework is depicted in Figure 5.5. The controller reacts on discrete system events $\tilde{e}_s$ produced by the generator or external events $\tilde{e}_r$, updates its internal states $\tilde{q}$ and passes new control symbols $\tilde{u}$ to the actuator. The actuator produces input signals $u(t)$ to the plant which in its turn affect the evolution of the continuous states $x(t)$.

We adopt the notation of [57] for distinguishing between signals and symbols and use tildes to indicate a symbol valued variable. For example, we denote by $x(t)$ (or simply x) a vector of continuous time signals, while $\tilde{x}$ is to be interpreted as a vector of symbols.

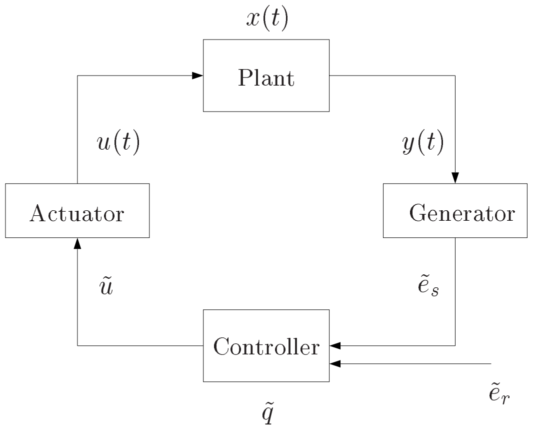


Figure 5.5: System configuration.

A more detailed description of the components in our framework follows.

5.2.1 Continuous Plant

The plant operates entirely in the continuous domain and is therefore conveniently modeled using, for instance, a nonlinear state space description. We find such a description, however, to general to allow any reasonable analysis and we therefore restrict the model structure.

Definition 5.5 (Plant)
The plant is continuous and time invariant, with dynamics piecewise affine in the continuous states and polynomial in the continuous inputs. More formally, we have

the plant dynamics given by

$$\dot{x}(t) = f(x(t), u(t)) = A(u(t))x(t) + b(u(t)) \tag{5.8}$$

$$y(t) = h(x(t)) = x(t) \tag{5.9}$$

where $x(t) \in \mathbb{R}^n$ is the continuous state vector, $u(t) \in \mathbb{R}^m$ is a vector of input signals and $y(t) \in \mathbb{R}^p$ is the output. In the sequel we will assume that all continuous states are measured, hence $y(t) = x(t)$. Furthermore, the elements of the input vector u enter the $n \times n$ matrix $A(u)$ and the $n \times m$ vector $b(u)$ polynomially.

Remark 5.5
Although we refer to the continuous part of the system as the plant, it may also contain some purely continuous controllers.

Note that when using (5.8) as a component in our framework for modeling switched systems, we are anticipating that the control signals $u(t)$ will be piecewise constant, yielding an affine state space model for each constant value, as described in Section 5.1.2. Without this assumption on the control signals, the model (5.8) is, although still correct, rather pointless.

5.2.2 DEDS Controller

The controller operates entirely in the discrete symbol domain and is modeled using the finite state machine framework.

Definition 5.6 (Controller)
The controller is purely discrete and has internal states. It is modeled as a deterministic state machine, $\mathcal{M}_C = \{\tilde{\mathcal{Q}}, \tilde{q}^0, \tilde{\mathcal{E}}, k, \tilde{\mathcal{U}}, l\}$ where $\tilde{\mathcal{Q}}$ is a set of controller states and $\tilde{q}_0 \in \tilde{\mathcal{Q}}$ is an initial state, $\tilde{\mathcal{E}}$ is a set of controller input symbols and $k : \tilde{\mathcal{Q}} \times \tilde{\mathcal{E}} \to \tilde{\mathcal{Q}}$ is the controller state transition relation. Furthermore, $\tilde{\mathcal{U}}$ is the set of controller output symbols and $l : \tilde{\mathcal{Q}} \to \tilde{\mathcal{U}}$ is the controller output function.

Note that $\mathcal{E}$ may contain both system and external events. We will distinguish between the two when necessary.

We see from the definition of the controller output function that the controller FSM is of Moore type. The implications of this are twofold. First, we are forced to introduce sufficiently many controller states to allow us to express the output in terms of the internal states only. But this is actually quite convenient, since the memory provided by the additional states serves as mechanism for keeping track of the dynamics associated with each region. Second, since the controller output is associated only with the actual state of the controller, we can consider it to be defined at all times. This is in contrast with the output of a Mealy machine which, from a modeling perspective, is more naturally viewed as a symbol defined only at the instances when the FSM makes a transition.

5.2.3 Actuator

We now define the part of the interface which transforms symbols to signals.

Definition 5.7 (Actuator)
The actuator is a mechanism for applying continuous control signals $u(t)$ in accordance to the discrete control symbols $\tilde{u}$. The most straightforward manner to accomplish this is translating the Boolean symbol `true` to the real constant 1 and the Boolean symbol `false` to the real constant 0,

$$\tilde{u}_i = \texttt{true} \Rightarrow u_i(t) = 1 \tag{5.10}$$

$$\tilde{u}_i = \texttt{false} \Rightarrow u_i(t) = 0 \tag{5.11}$$

where $1 \leq i \leq m$. This can be represented by the actuator function $\alpha : \mathbb{B}^m \to \mathbb{R}^m$ where

$$\alpha(\tilde{u}) = \{u \in \mathbb{R}^m \mid \bigwedge_{1 \leq i \leq m} \big[\tilde{u}_i \to [u_i = 1]\big] \wedge \big[\neg\tilde{u}_i \to [u_i = 0]\big]\} \tag{5.12}$$

Thus, for a given control symbol, $\tilde{u}^1$, we obtain constant system matrices

$$A(\tilde{u}^1) = A(\alpha(\tilde{u}^1)) \tag{5.13}$$

$$b(\tilde{u}^1) = b(\alpha(\tilde{u}^1)) \tag{5.14}$$

and as a result, the plant model describes an affine system for every control symbol.

5.2.4 Generator

Here we define the part of the interface which transforms signals into symbols, and provides the controller with discrete system events.

Definition 5.8 (Generator)
The generator is a mechanism for relating the discrete symbols $\tilde{e}$ to the continuous signals, $x(t)$. The discrete symbols are generated using a set of hyperplanes, $\mathcal{H}_i = \{x \mid c_i^T x - d_i = 0\}$. Events occur when the continuous trajectory enters or leaves a hyperplane, i.e., at time instances t_j where either $x(t_j^-) \notin \mathcal{H}_i$ and $x(t_j) \in \mathcal{H}_i$ or $x(t_j^-) \in \mathcal{H}_i$ and $x(t_j) \notin \mathcal{H}_i$. In connection with an event, an output symbol is generated according to

$$x(t_j) \in \mathcal{H}_i \Rightarrow \tilde{e}_i = \texttt{true} \tag{5.15}$$

$$x(t_j) \notin \mathcal{H}_i \Rightarrow \tilde{e}_i = \texttt{false} \tag{5.16}$$

Remark 5.6
Note that the actuator can be defined once and for all using 5.7, while the generator needs to be obtained for each specific application.

5.2.5 Switched Systems - A Definition

Let us summarize the framework presented in the previous sections by giving a formal definition of what we mean by a switched system.

Definition 5.9 (Switched System)
We refer to a switched system as a system given by a model

$$SwS = \{(A(u), b(u)), \mathcal{M}_C, \mathcal{H}\} \tag{5.17}$$

where $(A(u), b(u))$ are the system matrices of an affine state space model, $\mathcal{M}_C$ is a discrete event controller and $\mathcal{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_r\}$ is a set of hyperplanes defining a generator. Furthermore, the parameters of the system matrices are given by the actuator function.

As a result, given the three components of SwS, we now have a compact description (model) of a piecewise linear switched system.

5.2.6 A Simple Example

The above definitions are illustrated in the following example.

Example 5.5 (Water Tank, contd.) *Consider the controlled water tank in Figure 5.6. We assume that the tank is linear, i.e., we can obtain a linearization*

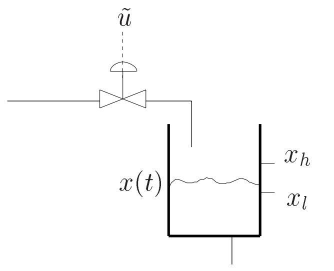


Figure 5.6: A water tank.

which is valid for the whole continuous region. This leads to the plant model

$$\dot{x} = ax + bu \tag{5.18}$$

where a, b are constants. In this one dimensional example, the generator consists of points on the real line; since we are here only interested in transitions from normal level to high or low level, the number of points are two,

$$\mathcal{H}_1 = \{x \in \mathbb{R} \mid x - x_l = 0\} \tag{5.19}$$
$$\mathcal{H}_2 = \{x \in \mathbb{R} \mid x - x_h = 0\} \tag{5.20}$$

Thus, the event $\tilde{e} = (\texttt{true}, \texttt{false})^T$ corresponds to passing the low level indicator (in either direction) while $\tilde{e} = (\texttt{false}, \texttt{true})^T$ represents going from normal level to high level or from high level to normal. The controller has the states on and off, represented by $\tilde{q} = \texttt{true}$ and $\tilde{q} = \texttt{false}$, respectively, and the controller input is the event vector $\tilde{e}$. The controller is designed to keep the level between high and low, this is accomplished using the transition relation defined by Table 5.3. The controller output is simply defined by $l(\tilde{q}) = \tilde{q}$ and we can select the initial

$k(\tilde{q}, \tilde{e})$		$\tilde{e}$: $\begin{pmatrix}\texttt{true}\\ \texttt{false}\end{pmatrix}$	$\tilde{e}$: $\begin{pmatrix}\texttt{false}\\ \texttt{true}\end{pmatrix}$
$\tilde{q}$	false	true	false
	true	true	false

Table 5.3: Controller transition function.

controller state as $\tilde{q}^0 = \texttt{true}$. The resulting controller FSM is shown in Figure 5.7, and a typical operation of the tank is visualized in Figure 5.8 where the level is plotted as a function of time and the evolution of the controller state is shown.

From Figure 5.8 we see that the constraints are only effective when hit from

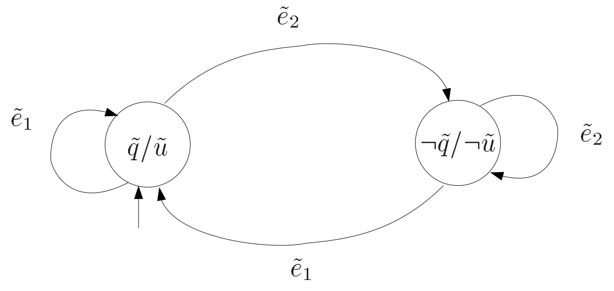


Figure 5.7: The controller finite state machine.

"inside", i.e., from the interval of normal operation. The constraints can thus be seen as "single sided". Usually, these types of constraints are utilized to implement hysteresis behavior; this is also the case in our example.

Note that in the above example, we introduced hysteresis by utilizing the discrete dynamics of the controller. The controller state thus provides the memory required to implement the hysteresis. This is always possible and provides a convenient way of modeling single sided constraints without having to explicitly keep track of in which direction a constraint is being crossed. In fact, the bookkeeping is performed for us by the state (memory) of the discrete controller.

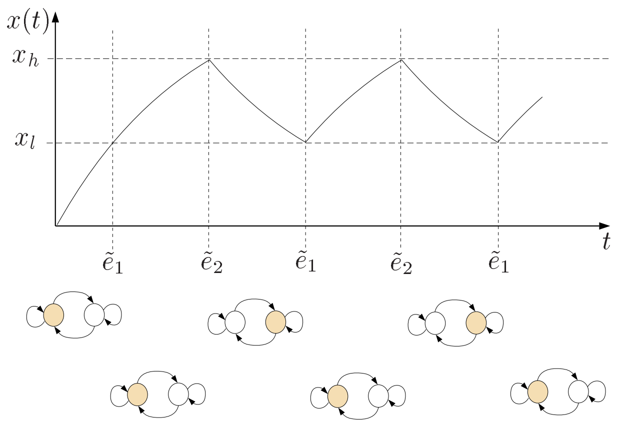


Figure 5.8: Visualization of tank operation. The upper part shows the level as a function of time while the lower part shows the controller state.

Part III

Analysis

6

Discrete Event Systems

6.1 Introduction

Analysis of a discrete event system consists of verifying properties of a model of the system. A property of a DEDS is an expression stated in some formalism which denotes a subset of the discrete state space. For instance, it can be a set of states that should never be entered or a set of specific sequences of states that should be executed. The process of obtaining guarantees for that the model has specific properties is referred to as formal verification.

A distinction is made between two basic approaches to discrete systems verification, the deductive or theorem proving approach and the algorithmic or model checking approach [12].

- Theorem Proving: Within the theorem proving approach, properties are inferred from mathematical axioms and proof rules concerning the dynamics of the system. The main disadvantage with this approach is that it is not automated. Even though the process can be aided by a computer program (theorem prover), it usually requires input from a human with expert knowledge about the dynamics of the system as well as the mechanisms of the theorem prover.

- Model Checking: In contrast to theorem proving, model checking is fully automated. Having obtained a system model and a specification of desired system properties, we feed these into a computer program (model checker) and expect it to return an answer without any further intervention.

We will mainly focus on the model checking approach throughout this thesis, therefore we will elaborate a little further on that topic.

Model checking can be seen as consisting of the following ingredients:

- A model of a discrete event system. This model should be in a format suitable for the available tools.
- A specification of properties to be examined. Again, the specification should be stated in a formalism which facilitates the use of computer tools.
- A model checking algorithm. It should be fully automated, i.e., given the model and specification it should arrive at a result without further interventions.

Furthermore, the result of a model checking algorithm consists either of the answer `true`, i.e., that the model satisfies the specification, or the value `false` along with a counterexample demonstrating why the specification is violated. This is of course extremely valuable for debugging design of complex systems.

Note that even though verification using the model checking approach is automated, manual work is required when producing the model and specifications. It is thus very important that the formalisms for these tasks are intuitive and easy to use. Furthermore, manual input may be needed to interpret the results from the model checker, e.g., in order to analyze counterexamples.

Basically, the model checking problem for discrete event systems is solved and current development is mainly focused on improving performance and applying the developed methods to practical applications. Several success stories concerning the use of model checking have appeared in recent years, especially for verifying digital hardware and protocols. For instance, in [15] model checking was used to detect errors in an IEEE standard, and in [37] a well established digital circuit is found erroneous through verification of a discrete circuit model obtained semi-automatically using conservative approximations. The main trend seems to be towards using symbolic representation of discrete models, implemented with the aid of Binary Decision Diagrams (BDDs) [7]. This allows efficient verification of quite complex systems [8], [22]. A nice orientation in the landscape of verification, with additional success stories, can be obtained from [12] and a comparative view of different verification approaches is provided in [23].

The chapter is organized as follows. In Section 6.2, we explain how a closed loop system is obtained from a discretized switched model and a discrete controller; this closed loop system is the object to be verified. In Section 6.3 we describe a formalism for specifying system properties, and in Section 6.4 we discuss the mechanisms of model checking algorithms.

6.2 The Model

In order to apply model checking, we need a model of the discrete event system. This model may have been obtained directly as a high level description of the system behavior, or, as we shall see in Chapter 9, as a discretization of the continuous part of a switched system. Hence, our treatment is given for a general DEDS modeled as a finite state machine and holds irrespective of the type of state machine treated.

We start by examining the effects of operating a discrete plant with a discrete controller, i.e., we seek an expression for the closed loop discrete event system.

In Section 5.1.1 we defined a finite state machine evolving according to a transition relation f and generating events as defined by an output function h,

$$f(x,u,x^+)\wedge[e_s = h(x,u)]. \tag{6.1}$$

The system is operated in closed loop with a DEDS controller which may have internal states and responds to system generated events e_s and external reference events e_r according to

$$k(q,e_s,e_r,q^+)\wedge[u = l(q)]. \tag{6.2}$$

As a result of this connection, we get a closed loop system defined by

$$f(x,u,x^+)\wedge k(q,e_s,e_r,q^+)\wedge[e_s = h(x,u)]\wedge[u = l(q)] \tag{6.3}$$

We see that we can write the output of the closed loop system in terms of combined plant and controller states, $\bar{x} = \{x,q\}$, and reference events, r, according to the closed loop output function

$$e_s = h(x,u) = h(x,l(q)) = \bar{h}(\bar{x}) \tag{6.4}$$

The formula (6.3) now becomes

$$\begin{aligned} f(x,l(q),x^+)\wedge k(q,\bar{h}(\bar{x}),e_r,q^+)\wedge[e_s = \bar{h}(\bar{x})] \\ = \bar{f}(\bar{x},e_r,\bar{x}^+)\wedge[e_s = \bar{h}(\bar{x})] \end{aligned} \tag{6.5}$$

which can be used as an underlying system model for the verification.

Having obtained a model of the closed loop system behavior, we now discuss a formalism for specifications of desired properties. This is the topic of the next section.

6.3 The Specification

In model checking of hardware and software systems, it is quite common to use temporal logic to specify system properties involving the concept of time. We will do that as well and it is therefore appropriate to give a brief introduction to this formalism.

Temporal logic is a formalism for introducing the concept of time in discrete event dynamic systems, see [50], [42] and [43]. However, time is not stated explicitly, but implicitly applying temporal operators to logical propositions, say p, stating for instance that "p always holds" or "eventually, p holds". These operators can be combined using the usual logical connectives and nested arbitrarily, thus providing quite powerful means of expressing temporal statements.

Several different temporal logics exist, each supplying its own operators, syntax and semantics. We will here use a logic referred to as computation tree logic or CTL [43], [23]. This logic is a subset of a more powerful logic called CTL* [23], [14]. The formulas in CTL* describe the properties of computation trees which basically are state transition structures which are unwound into an infinite tree starting from a designated initial state. The formulas are composed using path quantifiers describing the branching structure of the tree and temporal operators describing properties of the paths. In CTL, we make the restriction that each temporal operator needs to be preceeded by a path quantifier, thus all operators of CTL can be seen as branching-time operators.

There are two path quantifiers in CTL*, $\mathsf{A}(p)$ - "for all computation paths, p holds" and $\mathsf{E}(p)$ - "there exists a computation path where p holds". We will make use of four basic temporal operators,

- $\mathsf{F}(p)$ - future: "eventually, p holds"
- $\mathsf{G}(p)$ - globally: "always, p holds"
- $\mathsf{X}(p)$ - next: "in the next step, p holds"
- $\mathsf{U}(p, q)$ - until: "q eventually holds, until then, p holds"

Returning again to the subset CTL restricting to branching-time formulas, possible combinations of the above path quantifiers and temporal operators lead to the eight basic CTL operators listed in Table 6.1.

CTL Operator	Natural Language
$\mathsf{EF}(p)$	p can hold at some future step
$\mathsf{EG}(p)$	p can hold at all future steps
$\mathsf{EX}(p)$	p can hold in the next step
$\mathsf{EU}(p, q)$	q can hold at some future step, until then p will hold
$\mathsf{AF}(p)$	p must hold at some future step
$\mathsf{AG}(p)$	p must hold at all future steps
$\mathsf{AX}(p)$	p must hold in the next step
$\mathsf{AU}(p, q)$	q must hold at some future step, until then p will hold

Table 6.1: Basic CTL operators and their natural language interpretation.

Some examples of how we can use CTL formulas to specify properties of engineering systems are:

- The pump should always be on: $\mathsf{AG}(u_p)$.
- The temperature should never be above 200^o: $\neg\mathsf{EF}([T > 200]) = \mathsf{AG}([T \leq 200])$
- When the motor is turned off, the rotor should eventually stop: $\mathsf{AG}(\neg u_m \rightarrow \mathsf{AF}([\dot{x}_r = 0]))$

Let us now turn our attention to the actual model checking.

6.4 Model Checking

We now describe the mechanism of model checking. We start by stating the model checking problem and then discuss the use of fixed point iterations to perform the computations needed to solve the problem. We finally reflect on two different ways to implement model checking and discuss their relations to our specific framework.

6.4.1 Problem Statement

Assume that we are given a (closed loop) DEDS represented by a finite state machine. The behavior of such a system is then captured by its transition and output relations which are defined over a set of discrete state-, input- and output-variables. Let us denote the variables of such a system $z = \{x, u, y\}$ and the system model defining its behavior

$$M(z, z^+) = f(x, u, x^+) \wedge [y = h(x, u)] \tag{6.6}$$

Furthermore, we label a subset of the model state space as initial states, $\mathcal{I}$, defined implicitly by the system variables which satisfy a formula $I(z)$. Finally, we have a specification, either in terms of a set of bad states, $\mathcal{B}$, and/or a set of goal states, $\mathcal{G}$, or in terms of a CTL formula in the system variables which reflects our requirements on the system behavior. We will denote such a formula describing unwanted behavior $S_B(z)$ while a CTL formula expressing wanted properties is denoted $S_G(z)$.

Depending on the type of specifications given, we can perform the verification in two manners:

- Given a set of bad states, $\mathcal{B}$, and/or goal states, $\mathcal{G}$, obtain the set of states reachable from the initial states $\mathcal{I}$, call it $\mathcal{R}$. Now, the model checking consists of determining if

  $$\mathcal{B} \cap \mathcal{R} = \emptyset \tag{6.7}$$

  and/or

  $$\mathcal{G} \subseteq \mathcal{R}. \tag{6.8}$$

- Given a CTL formula, $S_B(z)$ or $S_G(z)$, obtain the set of states which satisfy the formula and denote it by $\mathcal{S}_B$ or $\mathcal{S}_G$, respectively. Now, the model checking consists of determining if
$$\mathcal{I} \cap \mathcal{S}_B = \emptyset \tag{6.9}$$
and/or
$$\mathcal{I} \subseteq \mathcal{S}_G. \tag{6.10}$$

Even though the above is formulated in terms of set operations, we see directly that the operations can be performed using the logical formulas defining the corresponding sets. We also see that both approaches involve the computation of certain sets, either the reachable set $\mathcal{R}$ or the specification sets $\mathcal{S}_B$ or $\mathcal{S}_G$. These computations involve evaluation of the dynamic properties of the system, described by the relation defined by $M(z, z^+)$, and thus need to be performed in a number of steps. Since all models are finite, this computation is bound to terminate, i.e., eventually the iterations involved will reach a fixed point.

6.4.2 Fixed Point Iterations

We will now describe how the reachable set and specification set can be obtained for a given relation $\mathcal{M}$.

We know that for sets $\mathcal{P} = \{z \mid P(z)\}$ and $\mathcal{Q} = \{z \mid Q(z)\}$, we can perform the usual set operations using the usual logical connectives, $\neg P(z)$, $P(z) \vee Q(z)$, and $P(z) \wedge Q(z)$, see Section 2.1. An additional interesting operation we can perform is, given a (transition) relation $\mathcal{M}$, we obtain the set of post-states related through $\mathcal{M}$ to pre-states in $\mathcal{P}$. We call the set of post-states the image of $\mathcal{P}$ with respect to $\mathcal{M}$. Of course, this operation can also be performed on the converse of $\mathcal{M}$, yielding the pre-states related to post-states in $\mathcal{P}$. We now define the operators on the indicator functions $P(z)$ and $M(z, z')$ which can be used to perform this kind of one step iteration of the relation.

Definition 6.1 (Forward and Backward Image)
Given a relation, $\mathcal{M} = \{(z, z') \mid M(z, z')\}$, and a set of states $\mathcal{P} = \{z \mid P(z)\}$, the forward image of $\mathcal{P}$ with respect to $\mathcal{M}$ is
$$\{z \mid \gamma_1^+(M(z, z'), P(z))\} \tag{6.11}$$
where
$$\gamma_1^+(M, P)(z) = \exists z'[P(z) \wedge M(z', z)] \tag{6.12}$$
Similarly, the backward image of $\mathcal{P}$ with respect to $\mathcal{M}$ is
$$\{z \mid \gamma_1^-(M(z, z'), P(z))\} \tag{6.13}$$
where
$$\gamma_1^-(M, P)(z) = \exists z'[P(z') \wedge M(z, z')] \tag{6.14}$$

The image operators enable us to extend our arsenal to operators involving next or previous step calculations, e.g., the set of states reachable in one transition or the set of states for which a property holds in the next step ($\mathsf{EX}(p)$). Of course, we can apply these operators repeatedly; this enables us to do calculations involving several steps. Let us start by looking at some properties of such repeated applications of set functions.

Definition 6.2 (Monotonic Functions and Fixed Points)
Let $\mathcal{P} = \{z \mid P(z)\}, \mathcal{Q} = \{z \mid Q(z)\}$ be sets and $\mathcal{T} : \mathcal{P} \to \mathcal{Q}$, $\mathcal{T} = \{z \mid \tau(z)\}$ be any function. We say that $\mathcal{T}$ is monotonic when

$$\mathcal{P} \subseteq \mathcal{Q} \implies \mathcal{T}(\mathcal{P}) \subseteq \mathcal{T}(\mathcal{Q}) \tag{6.15}$$

If $\mathcal{T}$ is monotonic and the universal set Ω is finite, the recursive computation

$$\emptyset \subseteq \mathcal{T}(\emptyset) \subseteq \mathcal{T}(\mathcal{T}(\emptyset)) \subseteq \cdots \subseteq \{z \mid \mu_P(\tau(P))(z)\} \tag{6.16}$$

terminates in at most $|\Omega|$ steps resulting in the least fixed point of $\mathcal{T}$, represented by $\mu_P(\tau(P))(z)$. Similarly, the computation

$$\Omega \supseteq \mathcal{T}(\Omega) \supseteq \mathcal{T}(\mathcal{T}(\Omega)) \supseteq \cdots \supseteq \{z \mid \nu_P(\tau(P))(z)\} \tag{6.17}$$

terminates in at most $|\Omega|$ steps and yields the greatest fixed point of $\mathcal{T}$ which is represented by $\nu_P(\tau(P))(z)$.

Note that the monotonicity condition can be expressed entirely in terms of the formulas defining $\mathcal{P}, \mathcal{Q}$ and $\mathcal{T}$, i.e., $\tau(z)$ defines a monotonic function provided that $P(z) \to Q(z)$ implies that $\tau(P(z)) \to \tau(Q(z))$. Accordingly, the subset chains can be formed using

$$\texttt{false} \to \tau(\texttt{false}) \to \tau(\tau(\texttt{false})) \to \cdots \to \mu_P(\tau(P))(z)$$

and

$$\texttt{true} \leftarrow \tau(\texttt{true}) \leftarrow \tau(\tau(\texttt{true})) \leftarrow \cdots \leftarrow \nu_P(\tau(P))(z)$$

We now have all the tools we need in order to compute both the reachable set, $\mathcal{R}$, and the specification set, $\mathcal{S}$. Starting with the former, we have the following definition.

Definition 6.3 (Reachable States)
Given a set of initial states $I(z)$ and a transition relation $M(z, z')$, the set of forward reachable states is defined by

$$R^+(z) = \mu_P(I(z) \vee \gamma_1^+(M, P)(z)) \tag{6.18}$$

Analogously, the set of backward reachable states is

$$R^-(z) = \nu_P(I(z) \vee \gamma_1^-(M, P)(z)) \tag{6.19}$$

In order to obtain the specification set, defined by a formula in CTL, we first need to characterize the various CTL operators.

Definition 6.4 (Characterization of CTL)
The CTL operators can be characterized as either images or fixed points with respect to the transition relation $M(z, z')$ using

$$\mathsf{EF}(p) = \mu_x(p \vee \mathsf{EX}(x)) \tag{6.20}$$
$$\mathsf{EG}(p) = \nu_x(p \wedge \mathsf{EX}(x)) \tag{6.21}$$
$$\mathsf{EX}(p) = \gamma_1^-(M, p) \tag{6.22}$$
$$\mathsf{EU}(p, q) = \mu_x(q \vee p \wedge \mathsf{EX}(x)) \tag{6.23}$$
$$\mathsf{AF}(p) = \mu_x(p \vee \mathsf{AX}(x)) \tag{6.24}$$
$$\mathsf{AG}(p) = \nu_x(p \wedge \mathsf{AX}(x)) \tag{6.25}$$
$$\mathsf{AX}(p) = \neg\mathsf{EX}(\neg p) \tag{6.26}$$
$$\mathsf{AU}(p, q) = \mu_x(q \vee p \wedge \mathsf{AX}(x)) \tag{6.27}$$

Having the means of obtaining the sets which satisfy a given CTL formula, the definition of a specification set is straightforward.

Definition 6.5 (Specification Sets)
Given a CTL specification, $S(z)$, and a transition relation, $M(z, z')$, the specification set is obtained as

$$\mathcal{S} = \{z \mid S(z)\} \tag{6.28}$$

where $S(z)$ is evaluated using the image and fixed point operations given by the characterization of CTL, together with the usual logical operations.

Of course all that has been said here holds irrespective of the type of specification, all we need to ensure is that we use the correct transition relation for each specification type.

Let us conclude this section on model checking by discussing briefly two implementation methods.

6.4.3 Explicit State Enumeration

The explicit state enumeration approach to performing model checking is probably the most intuitive one; it is also the method used in the earliest implementations of model checking tools [43], [13].

Basically, one performs an exhaustive search in the state space and constructs either the reachability set or the specification set. For instance, starting from the set of initial states, we use the transition relation to obtain the set of states reachable in one transition, i.e., the forward image. From that set we then use the transition relation again to obtain the set of states reachable from those sets etc.. The reachability graph obtained may be stored during the procedure using a suitable data structure as, for instance, adjacency lists. The procedure continues until the set of reachable states ceases to increase, that is, when we reach a fixed point. A similar procedure can of course be performed backwards or for a CTL formula in order to obtain the set of states satisfying a specification.

One realizes that the main drawback of performing an explicit traverse of the reachability graph is complexity; we soon hit inevitable time/memory bounds as we proceed through the state space and our data structure increases. However, this method can be used for verification of non-trivial examples and we will apply it in one of our examples in Chapter 11.

6.4.4 Symbolic Model Checking

In efforts to circumvent the complexity problems associated with implementations based on explicit state enumeration, McMillan [43] realized that performance could be increased dramatically by representing the Boolean formulas representing states and transition relations using Binary Decision Diagrams (BDDs) [7]. The fixed point iterations can then be performed by operating on the BDD representation.

Due to the implicit representation of states and transition relations via Boolean formulas, the symbolic approach often circumvents the complexity barriers of explicit methods. Furthermore, the BDD representation of the Boolean formulas allows for extremely efficient implementation of symbolic model checking. And although the worst case complexity of symbolic methods is no better than of explicit methods, in practice they perform much better. As a result, symbolic methods have extended considerably the sizes of problems which can be verified automatically from a given model and specification [23].

7

Continuous Systems

7.1 Introduction

Analysis of continuous systems is a somewhat more diverse field than that of discrete event systems. We will focus on analysis methods which are in their basic nature related to those for discrete event systems and where the combination of the two will enable us to treat mode switching systems.

In a manner similar to analysis methods for discrete event systems, we are interested in analyzing how the system will behave given some initial or final state(s). And we will also here be concerned both with avoiding certain states as well as ensuring that certain states will be reached.

The chapter is organized as follows. In Section 7.2 we define some properties of a contionuous time dynamics system, such as equilibrium, reachable set and region of attraction. In Section 7.3 we look at transition conditions for a dynamic system, i.e., conditions for the trajectory of a dynamic system to enter a subset of the state space. Finally, in Section 7.4, we look at how we can guarantee that approximations of reachable sets and regions of attraction are conservative, meaning that they can be used for determining transition conditions.

7.2 Dynamic Systems

Let the system dynamics be governed by the autonomous vector differential equation

$$\dot{x}(t) = f(x(t)),\ t \geq 0 \tag{7.1}$$

where $x(t) \in \mathbb{R}^n$ is the system state at time t and $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous. We then know that (7.1) has a unique solution corresponding to each initial state $x(0) = x_0$ [62] and denote the solution at time t by $s(t, x_0)$. The union of all solutions from time $t = 0$ to time t for a given initial state is referred to as a (solution) trajectory at time t and we denote it by $s[t, x_0] = \{x \in \mathbb{R}^n \mid x = s(\tau, x_0)\ ,\ \tau \in [0, t]\}$. The trajectory of a system is the set of all solutions obtained at any time, i.e., $s[x_0] = \{x \in \mathbb{R}^n \mid x = s(\tau, x_0)\ ,\ \tau \geq 0\}$.

7.2.1 Equilibrium

Now assume that the vector 0 is a unique equilibrium of the system (7.1), that is $f(0) = 0$ and $f(x) \neq 0\ ,\ \forall x \neq 0$. Let us look at some properties of the equilibrium.

Definition 7.1 (Equilibrium)
The equilibrium 0 is

i) stable if, for each $\epsilon > 0$, there exists a $\delta = \delta(\epsilon)$ such that

$$\|x_0\| < \delta(\epsilon) \implies \|s(t, x_0)\| < \epsilon,\ \forall t \geq 0. \tag{7.2}$$

ii) unstable if it is not stable

iii) attractive if there is a number $\eta > 0$ such that

$$\|x_0\| < \eta \implies s(t, x_0) \to 0 \text{ as } t \to \infty \tag{7.3}$$

iv) asymptotically stable if it is stable and attractive.

Definition 7.1.iii) states that solution trajectories starting sufficiently close to the origin will eventually approach it as time goes to infinity. This implies that there exists a set such that every trajectory starting from within that set will approach 0 as $t \to \infty$.

Definition 7.2 (Domain of Attraction)
Let 0 be the attractive equilibrium of the system (7.1). The domain of attraction $\mathcal{D}(0)$ is defined as

$$\mathcal{D}(0) = \{x_0 \in \mathbb{R}^n \mid s(t, x_0) \to 0 \text{ as } t \to \infty\} \tag{7.4}$$

We have considered the solution to a dynamic system when the system starts from a specific point in the state space and possibly converges to another point. We now move on to similar concepts when dealing with continuous sets rather than single points. This leads us to reachable sets and regions of attraction.

7.2.2 Reachable Set

We will frequently deal with the case where the initial state is not specified directly but given as a member of an initial set, $x_0 \in \mathcal{X}_0$. Hence, we are interested in the union of all solutions to (7.1) as the initial state varies over the set of initial states.

Definition 7.3 (Reach set, Reachable sets)
At a given time instance, the reach set at time t is

$$\mathcal{R}(t, \mathcal{X}_0) = \{x \in \mathbb{R}^n \mid x = s(t, x_0)\ ,\ x_0 \in \mathcal{X}_0\}. \tag{7.5}$$

Analogously to the case of single solutions, the union of all reach sets from time $t = 0$ to time t is

$$\mathcal{R}[t, \mathcal{X}_0] = \{x \in \mathbb{R}^n \mid x \in \mathcal{R}(\tau, \mathcal{X}_0)\ ,\ \tau \in [0, t]\} \tag{7.6}$$

and referred to as the reachable set at time t. The union of all reach sets

$$\mathcal{R}[\mathcal{X}_0] = \{x \in \mathbb{R}^n \mid x \in \mathcal{R}(\tau, \mathcal{X}_0)\ ,\ \tau \geq 0\} \tag{7.7}$$

is called the reachable set of system (7.1).

Remark 7.1
Other terminology for the reachable set of a dynamic systems exists, such as attainability domain, trajectory assembly/bundle or solution tube, see e.g. [ref t ex Varayia, Kurshanski]. Although the term reachable implies the existence of an external input that can be selected to manipulate the system trajectories, we adopt this nomenclature for the set of solutions obtained for a set of given initial states and a given, possibly infinite, time interval.

7.2.3 Region of Attraction

We now assume that we are given a set of goal states, $\mathcal{X}_f$ and wish to examine from which parts of the state space we can guarantee that solutions starting there will enter $\mathcal{X}_f$.

Definition 7.4 (Region of Attraction)
The region of attraction for a set $\mathcal{X}_f \subseteq \mathbb{R}^n$ is the set

$$\mathcal{A}[\mathcal{X}_f] = \{x_0 \in \mathbb{R}^n \mid \exists t \geq 0 \quad s.t. \quad s(t, x_0) \in \mathcal{X}_f\}. \tag{7.8}$$

Remark 7.2
Note that the region of attraction of a set as defined in Definition 7.4 is not the same concept as the domain of attraction of a limit set.

Remark 7.3
The region of attraction for a set of goal states $\mathcal{X}_f$ can also be seen as the backward reachable set taking $\mathcal{X}_f$ as a set of initial states, i.e.,

$$\mathcal{A}[\mathcal{X}_f] = \mathcal{R}^-[\mathcal{X}_f] = \{x \in \mathbb{R}^n \mid x = s^-(t, x_0) \ , \ x_0 \in \mathcal{X}_f \ , \ t \geq 0\} \tag{7.9}$$

where $s^-(t, x_0)$ is the solution to $\dot{x} = -f(x)$, $t \geq 0$, i.e., the solution obtained by starting in $\mathcal{X}_f$ and solving (7.1) backwards in time.

Example 7.1 *Consider the linear dynamic system*

$$\dot{x}(t) = Ax(t) = \begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix} x(t) \tag{7.10}$$

with the initial conditions confined to the unit circle centered at $(2, 3)^T$, that is

$$\mathcal{X}_0 = \{x_0 \in \mathbb{R}^2 \mid (x_{0,1} - 2)^2 + (x_{0,2} - 3)^2 \leq 1\} \tag{7.11}$$

The solution for a specific initial state x_0 is given by

$$s(t, x_0) = \begin{pmatrix} e^{-2t} & 0 \\ 0 & e^{-t} \end{pmatrix} x_0 \tag{7.12}$$

and hence the reachable set is obtained as

$$\begin{aligned} \mathcal{R}[\mathcal{X}_0] &= \{x \in \mathbb{R}^2 \mid x = s(t, x_0) \ , \ x_0 \in \mathcal{X}_0 \ , \ t \geq 0\} \\ &= \{x \in \mathbb{R}^2 \mid x_1 = e^{-2t} x_{0,1} \ , \ x_2 = e^{-t} x_{0,2} \ , \\ &\qquad (x_{0,1} - 2)^2 + (x_{0,2} - 3)^2 \leq 1 \ , \ t \geq 0\} \end{aligned} \tag{7.13}$$

Here we can eliminate the time t, this leads to

$$\begin{aligned} \mathcal{R}[\mathcal{X}_0] = \{x \in \mathbb{R}^2 \mid x_{0,2}^2 x_1 = x_{0,1} x_2^2 \ , \ (x_{0,1} - 2)^2 + (x_{0,2} - 3)^2 \leq 1 \ , \\ x_1 \leq x_{0,2} \ , \ x_2 \leq x_{0,2}\} \end{aligned} \tag{7.14}$$

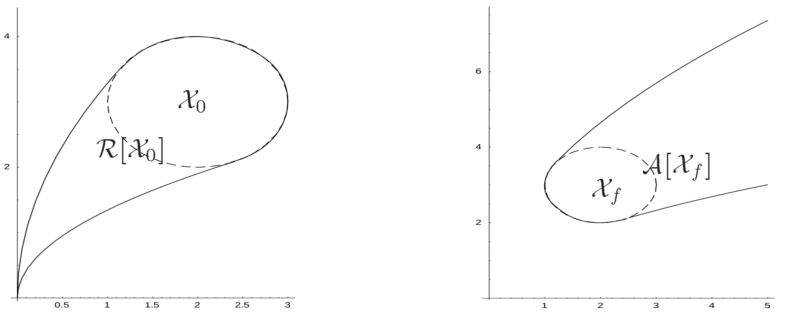
which can be rewritten as

$$\begin{aligned} \mathcal{R}[\mathcal{X}_0] = \{x \in \mathbb{R}^2 \mid f_1(x_1, x_2) \leq 0 \wedge x_2 \leq a x_1 + b \ \vee \\ f_2(x_1, x_2) \leq 0 \wedge x_2 \geq a x_1 + b\} \end{aligned} \tag{7.15}$$

where $f_1(x_1, x_2) = x_2^8 - 28 x_1 x_2^6 + 423 x_1^2 x_2^4 - 2944 x_1^3 x_2^2 + 4096 x_1^4$, $f_2(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2 - 1$, and $x_2 = a x_1 + b$ is the line going through the points obtained as the real solutions to $f_1(x_1, x_2) = f_2(x_1, x_2) = 0$.
The region of attraction can be obtained in a similar fashion by substituting $-A$ for A in (7.10), that is by finding the set reachable backwards in time. This results in

$$\begin{aligned} \mathcal{A}[\mathcal{X}_f] = \{x \in \mathbb{R}^2 \mid f_1(x_1, x_2) \leq 0 \wedge x_2 \geq a x_1 + b \ \vee \\ f_2(x_1, x_2) \leq 0 \wedge x_2 \leq a x_1 + b\} \end{aligned} \tag{7.16}$$

The sets $\mathcal{R}[\mathcal{X}_0]$ and $\mathcal{A}[\mathcal{X}_f]$ are illustrated in Figure 7.1.



(a) Reachable set. (b) Domain of attraction.

Figure 7.1: Illustration of the sets obtained in Example 7.1.

7.3 Transition Conditions

We will now consider conditions for the solution to (7.1) to enter a set $\mathcal{X}_f \subseteq \mathbb{R}^n$. We are interested in two kinds of conditions, namely, starting from an initial set $\mathcal{X}_0 \subseteq \mathbb{R}^n$,

- when is it possible that a solution will enter $\mathcal{X}_f$?
- when can we guarantee that all solutions will enter $\mathcal{X}_f$?

When a solution enters $\mathcal{X}_f$ at some time $t \geq 0$, we say that a transition is made to the set $\mathcal{X}_f$. Addressing the first condition for a transition we have the following lemma.

Lemma 7.1
Let $\mathcal{X}_0 \subseteq \mathbb{R}^n$ be a set of initial states and $\mathcal{R}[\mathcal{X}_0]$ the reachable set. Then a transition to a set $\mathcal{X}_f \subseteq \mathbb{R}^n$ is possible if and only if

$$\mathcal{X}_f \cap \mathcal{R}[\mathcal{X}_0] \neq \emptyset \tag{7.17}$$

Proof If the intersection of $\mathcal{X}_f$ and $\mathcal{R}[\mathcal{X}_0]$ is nonempty, then there exists a point $x_0 \in \mathcal{X}_0$ and a $t \geq 0$ such that $s(t, x_0) \in \mathcal{X}_f$, hence a transition is made possible by selecting x_0 as an initial state. Conversely, if the intersection is empty no such point exists and no transition is possible. $\square$

Similarly, we can address the second condition where we wish to guarantee that a transition will be made to a set of goal states.

Lemma 7.2
Let $\mathcal{A}[\mathcal{X}_f]$ be the region of attraction for a set $\mathcal{X}_f \subseteq \mathbb{R}^n$. Then a transition to $\mathcal{X}_f$ is guaranteed if and only if

$$\mathcal{X}_0 \subseteq \mathcal{A}[\mathcal{X}_f] \tag{7.18}$$

where $\mathcal{X}_0 \subseteq \mathbb{R}^n$ is the set of initial states.

Proof If $\mathcal{X}_0$ is a subset of the region of attraction, then we have that for all $x_0 \in \mathcal{X}_0$ there exists a $t \geq 0$ such that $s(t, x_0) \in \mathcal{X}_f$, hence we will have a transition to $\mathcal{X}_f$ irrespective of the choice of x_0. However, if $\mathcal{X}_0 \nsubseteq \mathcal{A}[\mathcal{X}_f]$, then there exists an initial state $x_0 \in \mathcal{X}_0$ such that the solution $s(t, x_0)$ never enters $\mathcal{X}_f$. Since we cannot exclude this initial state we cannot guarantee a transition. □

Corollary 7.1
Let $\mathcal{X}_0 \subseteq \mathbb{R}^n$ be a set of initial states and $\mathcal{R}[\mathcal{X}_0]$ the reachable set. Furthermore, let $\hat{\mathcal{R}}[\mathcal{X}_0] \supseteq \mathcal{R}[\mathcal{X}_0]$ be an outer approximation of $\mathcal{R}[\mathcal{X}_0]$. Then no transition to the set $\mathcal{X}_f \subseteq \mathbb{R}^n$ is possible if

$$\mathcal{X}_f \cap \hat{\mathcal{R}}[\mathcal{X}_0] = \emptyset \tag{7.19}$$

Proof We have that $\emptyset = \mathcal{X}_f \cap \hat{\mathcal{R}}[\mathcal{X}_0] \supseteq \mathcal{X}_f \cap \mathcal{R}[\mathcal{X}_0] \Rightarrow \mathcal{X}_f \cap \mathcal{R}[\mathcal{X}_0] = \emptyset$ and the necessary condition of Lemma 7.1 is violated. □

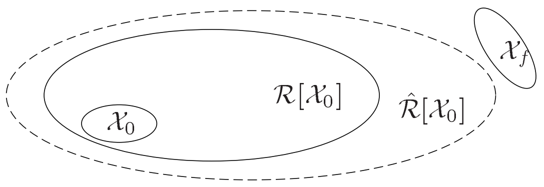

$\mathcal{X}_f$
$\mathcal{R}[\mathcal{X}_0]$
$\hat{\mathcal{R}}[\mathcal{X}_0]$
$\mathcal{X}_0$

Figure 7.2: An illustration of Corollary 7.1

Corollary 7.2
Let $\mathcal{A}[\mathcal{X}_f]$ be the region of attraction for a set $\mathcal{X}_f \subseteq \mathbb{R}^n$. Furthermore, let $\hat{\mathcal{A}}[\mathcal{X}_f] \subseteq \mathcal{A}[\mathcal{X}_f]$ be an inner approximation of $\mathcal{A}[\mathcal{X}_f]$. Then a transition to $\mathcal{X}_f$ is guaranteed if

$$\mathcal{X}_0 \subseteq \hat{\mathcal{A}}[\mathcal{X}_f] \tag{7.20}$$

where $\mathcal{X}_0 \subseteq \mathbb{R}^n$ is the set of initial states.

Proof We have that $\mathcal{X}_0 \subseteq \hat{\mathcal{A}}[\mathcal{X}_f] \subseteq \mathcal{A}[\mathcal{X}_f]$ and hence the sufficient condition of Lemma 7.2 is fulfilled. □

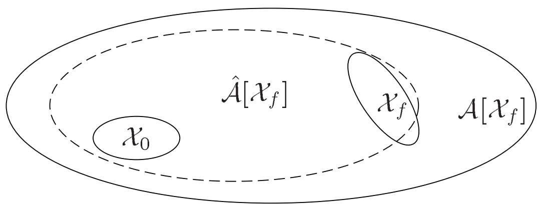

$\hat{\mathcal{A}}[\mathcal{X}_f]$
$\mathcal{X}_f$
$\mathcal{A}[\mathcal{X}_f]$
$\mathcal{X}_0$

Figure 7.3: An illustration of Corollary 7.2

7.4 Conservative Approximations

In this section we will develop conditions that enable us to guarantee that approximations of reachable sets and regions of attraction are conservative, that is outer and inner approximations of $\mathcal{R}[\mathcal{X}_0]$ and $\mathcal{A}[\mathcal{X}_f]$, respectively. This enables the use of the approximations with Corollaries 7.1 and 7.2 to determine if transitions to specific sets are possible or can be guaranteed. Having obtained these conditions, we can use them for computing approximating sets in a constructive manner, that is we will choose sets with a specific structure which facilitates computation and representation. It turns out that the power cones of Chapter 4 fall into this category.

An important concept for the development of approximation methods is the one of sets with the property that once the state trajectory enters the set it will remain in it for all future times.

Definition 7.5 (Invariant set)
A set $\mathcal{S} \subseteq \mathbb{R}^n$ is called an invariant set of the differential equation (7.1) if

$$x_0 \in \mathcal{S} \implies s(t, x_0) \in \mathcal{S},\ \forall t > 0. \tag{7.21}$$

We can now state conditions for conservative approximations of reachable sets and regions of attractions.

Lemma 7.3
Let $\mathcal{S}$ be an invariant set of the system (7.1) and $\mathcal{X}_0$ a set of initial states such that $\mathcal{X}_0 \subseteq \mathcal{S}$. Then $\mathcal{R}[\mathcal{X}_0] \subseteq \mathcal{S}$ and we can take $\hat{\mathcal{R}}[\mathcal{X}_0] = S$ as an outer approximation of $\mathcal{R}[\mathcal{X}_0]$.

Proof Follows directly from the definition of invariant sets and that $\mathcal{X}_0 \subseteq \mathcal{S}$. □

Lemma 7.4
Let $\mathcal{S}$ be an invariant set of the system (7.1) and $\mathcal{X}_f$ a set of goal states such that $\mathcal{S} \cap \mathcal{X}_f \setminus \partial\mathcal{X}_f \neq \emptyset$. Assume that there exists a C^1 function $V : \mathbb{R}^n \to \mathbb{R}$ such that

(i) $V(x) > 0\ ,\ \dot{V}(x) < 0\ ,\ \forall x \in \mathcal{S} \setminus (\mathcal{X}_f \setminus \partial\mathcal{X}_f)$,

(ii) $\exists x_f \in \mathcal{S} \cap \mathcal{X}_f \setminus \partial\mathcal{X}_f$ *s.t.* $V(x_f) = 0$,

(iii) $\forall i \in \{1..n\}$, $|x_i| \to \infty \;\Rightarrow\;$ *either* $V(x) \to \infty$ *or* $x \notin \mathcal{S}$.

Then $\mathcal{S} \subseteq \mathcal{A}[\mathcal{X}_f]$ *and we can take* $\hat{\mathcal{A}}[\mathcal{X}_f] = \mathcal{S}$ *as an inner approximation of* $\mathcal{A}[\mathcal{X}_f]$.

Proof We show that every solution starting in $\mathcal{S}$ will reach the boundary of $\mathcal{X}_f$ in finite time.

Since $\mathcal{S}$ is invariant, we know that if $x_0 \in \mathcal{S}$, then $s(t, x_0) \in \mathcal{S}$, $\forall t \geq 0$. Furthermore, even though $\mathcal{S}$ can be unbounded, conditions (i) and (iii) ensure that given any initial condition $x_0 \in \mathcal{S}$, the trajectory will not escape to infinity but remain in the bounded region $\mathcal{S} \cap \{x \in \mathbb{R}^n \mid V(x) \leq V(x_0)\}$.

Finally, we show by contradiction that all trajectories converge asymptotically to a point in $\mathcal{S} \cap \mathcal{X}_f \setminus \partial\mathcal{X}_f$. Since V is lower bounded and decreases continually, V tends to a limit δ_1 such that $V(s(t, x_0)) \geq \delta_1$, $\forall t \geq 0$. Assume that this limit is not zero, i.e., that $\delta_1 > 0$. Then, since V is continuous and $V(x_f) = 0$, we have that for every $x_f \in \mathcal{S} \cap \mathcal{X}_f \setminus \partial\mathcal{X}_f$ there exists a neighborhood $\|x - x_f\| < \epsilon$ that the trajectory never enters. But, since $\dot{V}$ is continuous and negative definite, we have $\dot{V} \leq -\delta_2 < 0$. This is a contradiction, because it would imply that $V(t)$ decreases from its initial value $V(x_0)$ to a value smaller than δ_1 in a finite time smaller than $\frac{V(x_0)-\delta_1}{\delta_2}$. Hence, every trajectory starting in $\mathcal{S}$ converges asymptotically to a point $x_f \in \mathcal{S} \cap \mathcal{X}_f \setminus \partial\mathcal{X}_f$ and there will exist a finite time t_f such that $s(t_f, x_0) \in \partial\mathcal{X}_f$. □

Both the conditions of Lemma 7.3 and Lemma 7.4 require the computation of an invariant set for the system (7.1). A sufficient condition for a set to be invariant is that the vector field is directed into the set at all points on the boundary. When the set is described as a sublevel set of a function $V : \mathbb{R}^n \to \mathbb{R}$, this condition can be formulated as

$$V_x(x) f(x) < 0 \quad , \quad \forall x \quad \text{s.t.} \quad V(x) = c \tag{7.22}$$

where $V_x(x)$ denotes the gradient of $V(x)$.

8

Switched Systems

8.1 Introduction

A verification method for hybrid systems should facilitate reasoning in the discrete domain while simultaneously ensuring the behavior in the continuous domain. This can be accomplished by abstracting from the continuous behavior in each mode of the switched system, replacing the detailed behavior in the mode by knowledge about what effect it will have on the discrete behavior. That basically means that one is only concerned with finding information about discrete transitions resulting from the continuous dynamics. Ideally, this would allow us to replace the switched system with a discrete event equivalent one and hence to reason about the discrete behavior.

Before looking at the details of verification of switched systems, let us first recall our definition of a switched system from Section 5.2. We have a system consisting of a continuous plant, described by the set of first order differential equations $\dot{x}(t) = A(u(t))x(t) + b(u(t))$ and a discrete event controller. Each mode i of the controller results in a control output $u(t)$ which is constant while the controller remains in that mode, $u(t) = u_i$. As a result, we can associate to each mode a time invariant affine model, $\dot{x}(t) = A_i x(t) + b_i$. The controller can change modes, say from mode i to mode j, as a result of events generated when the state trajectory, $x(t)$, hits a transition surface $\mathcal{T}_{ij}$. Here, the transition surfaces are hyperplanes.

We will in this chapter take a look at procedures for performing verification using the model checking approach. In Section 8.2 we look at how model checking is performed in the ideal situation when all sets involved can be computed and represented in an exact manner. In Section 8.3 we discuss how we deal with the fact that we need to resort to conservative approximations in order to have computational methods for performing the verification.

8.2 Idealized Model Checking

We saw in Lemmas 7.1 and 7.2 of Section 7.3 that reachable sets and regions of attraction are keys to determining whether a transition to a specific set is possible or can be guaranteed to occur, respectively. Hence, these sets can be used as the mechanism for determining the occurance of transitions to other modes through a transition surface. Since these sets in general are not easily computed and represented, we need to resort to sets that serve as conservative approximations and are more easily manipulated. However, for the sake of clarity it is useful to state an idealized verification procedure in order to avoid loosing track of the basic concepts involved.

8.2.1 Bad States

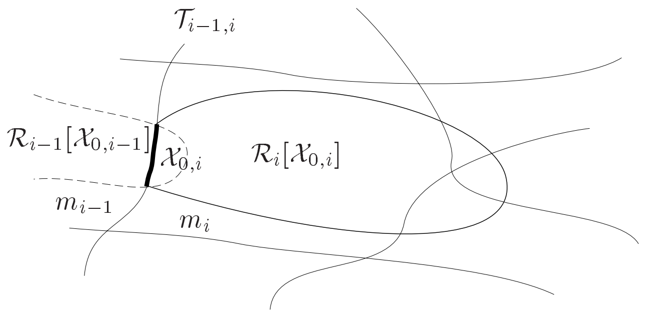
In a fashion similar to verification of purely discrete systems, we want to determine automatically if, given a model and a set of initial states, we run at the risk of entering certain specified "bad states".

Algorithm 8.1 (Model Checking - Bad States)

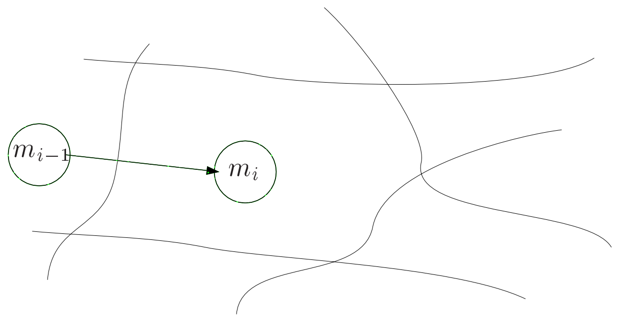
Given: A hybrid model, initial conditions and a specification of states to be avoided.
Sought: A yes/no answer to the question whether the model satisfies the specification for the given initial conditions. Furthermore, in the case of a negative answer, a counterexample.

1. *Compute the reachable set for the set of initial states, $\mathcal{R}_i[\mathcal{X}_0]$. Add the transition surfaces $\mathcal{T}_{ij}$ which are intersected to the list of surfaces. Add the reachable set to the set reachable from other initial states in the mode.*
2. *If no new modes are entered (list of surfaces is empty), backtrack to the previous mode. If there is no previous mode, exit with a positive answer, otherwise repeat this step for the previous mode.*
3. *Pick one surface out of the list of transition surfaces. Extend the mode automaton and check if any bad state has been reached. If so, exit with a negative answer and the counterexample.*
4. *Compute the initial set $\mathcal{X}_0$ for the next mode as the intersection of the reachable set and the transition surface. If this initial set is a subset of previously computed reachable set for that mode, goto step 2. Else, set current mode to next mode and go to step 1.*

In Figures 8.1 and 8.2 we illustrate the reachable set computation for a mode m_i and how the intersection with transition surfaces is used to extend the automaton and obtain new initial sets for the following modes. We see from Figure 8.2, where both mode m_{i+1} and mode m_{i+2} could be entered from mode m_i, that we may end up with a non-deterministic discrete abstraction of the original switched system. This happens even though we have not made any approximation, we merely view the switched system at a higher level of abstraction.


$\mathcal{T}_{i-1,i}$
$\mathcal{R}_{i-1}[\mathcal{X}_{0,i-1}]$
$\mathcal{X}_{0,i}$
$\mathcal{R}_i[\mathcal{X}_{0,i}]$
m_{i-1}
m_i

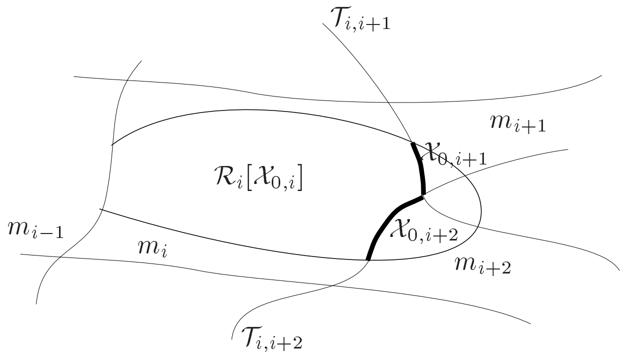
(a) Computation of reachable set.



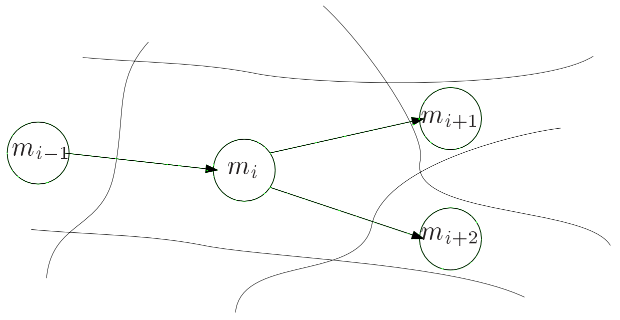
(b) The current mode automaton.

Figure 8.1: Visualization of Algorithm 8.1.

Note that Algorithm 8.1 is not guaranteed to terminate. For instance, we may get cyclic behavior where a transition is made to a previously visited mode, adding one point to the initial set of the mode. Hence, the reachable set would be slightly larger and the process might be repeated infinitely often.



(a) Intersection of reachable set with mode transition surfaces.



(b) The extended mode automaton.

Figure 8.2: Visualization of Algorithm 8.1

8.2.2 Goal States

The basic problem we formulate here is to determine automatically, given a model and a set of initial states, whether we can guarantee that we end up in a set of specified "goal states".

Algorithm 8.2 (Model Checking - Goal States)

Given: A hybrid model, final conditions and a specification of states to be attained.
Sought: A yes/no answer to the question whether the model satisfies the specification for the given final conditions. Furthermore, in the case of a negative answer, a counterexample.

1. *Compute the region of attraction for the set of final states, $\mathcal{A}_j[\mathcal{X}_f]$. Add the*

transition surfaces $\mathcal{T}_{ij}$ which are intersected to the list of surfaces. Add the region of attraction to the region of attraction found for other final states in the mode.

2. *If no new modes are found (list of surfaces is empty), backtrack to the previous mode. If there is no previous mode, exit with a negative answer and a counterexample, otherwise repeat this step for the previous mode.*
3. *Pick one surface out of the list of transition surfaces. Extend the mode automaton and check if all initial states are represented. If so, exit with a positive answer.*
4. *Compute the final set $\mathcal{X}_f$ for the next mode as the intersection of the region of attraction and the transition surface and add it to the list of initial sets. If this final set is a subset of a previously computed region of attraction for that mode, goto step 2. Else, set current mode to next mode and go to step 1.*

In step 4 of Algorithm 8.2 we add every new final set to the list of initial sets given as input to the algorithm. This is due to the fact that if the region of attraction for a final set in a mode intersects several transition surfaces, some trajectories might hit one of those surfaces before reaching the final set and hence cause the system to change mode. We therefore need to make sure that these trajectories end up in the goal states as well, this is accomplished by treating them as any other initial sets.

8.3 Model Checking using Approximations

Since we in general are unable to compute and represent the reachable sets and regions of attraction for a dynamic system, we need to resort to approximations of these sets. Furthermore, we may need to approximate the initial and final sets, e.g., in order to facilitate efficient computations or due to properties of the approximating sets.

The use of approximations does not alter the basic structure of the model checking algorithms. What changes is the interpretation of the result, e.g., a negative answer when checking if bad states are reached may not necessarily imply that the specification is violated and those states will be reached by the underlying system. In addition, since we replace initial and final sets by their approximations, we may need to introduce intermediary steps where these approximations are computed. The main concern is that these intermediate steps can be automated so that the verification procedure is fully algorithmic in the spirit of model checking tools for discrete event systems. This has been kept as a guideline throughout the development in this thesis.

We take two different approximation approaches. Common to both is that the transition hyperplanes define the boundaries of each mode and every cell in $\mathbb{R}^n$ created that way can thus be considered as corresponding to a discrete state in a discrete abstraction of the switched system. It then remains to determine the transition conditions, this is where the methods differ due to properties of the methods they use for approximating reachable sets and regions of attraction.

The first one, treated in Chapter 9, uses simple conditions on the vector fields at the boundaries of each mode to check if transitions are possible through them. This can be seen as approximating the reachable set for that mode with the entire cell defined by its transition hyperplanes. It also utilizes conditions on the vector field within the cell in order to determine if one or more of the boundaries has the whole cell as a region of attraction. Again this is a very conservative approximation and does not utilize at all the fact that the initial set of each mode is a subset of its cell. Although rough, this methods turns out to be quite useful for some systems, more specifically systems with real eigenvalues where the curvature of the trajectories is large compared with the geometry of the cells. This is demonstrated when the method is applied to two examples in Chapter 11.

The second method, which is described in Chapter 10, utilizes more structure, and is therefore likely to perform better in situations where the former method fails. Here we make use of the power cones devised in Chapter 4 for approximating reachable sets and regions of attraction. We preserve information about the initial set when a new mode is entered and use the fact that we can always find power cones that are invariant for the type of systems considered. Hence, the resulting approximations are reasonably tight.

Both methods are fully automated and rely on efficient computational tools. The former uses on linear programming and in fact there we also have the possibility of using symbolic computational procedures. The latter relies mainly on convex optimization for which there exist whery efficient tools. Both methods scale well up to higher dimensions, the main obstacle being the curse of dimensionality resulting in a dramatic increase in number of cells as the dimension increases. However, the fact that many engineering systems only reach a restricted subset of the modes that potentially could exist, can very well lead to that these larger dimensional systems can be verified as well.

We now discuss these two methods in more detail in the next two chapters.

9

Approximations using Polytopes

9.1 Introduction

We will now design a method for transforming a switched system with piecewise linear dynamics into different kinds of discrete event dynamical systems in a manner which allows us to draw conclusions about the behavior of the switched system, using the discrete model.

The translation to a DEDS involves three main steps, namely

- defining a discrete state space,
- defining a transition relation for transitions between discrete states, and
- defining an output function for the resulting finite state machine.

Defining an abstracted discrete state space and an output function turns out to be rather straightforward, while we will see that obtaining one transition relation which is a "good" abstraction of the switched dynamics is difficult. By a "good" abstraction, we mean one that enables us to use the discrete system obtained to check whether we can guarantee that either

- certain states will not be reached,
- certain states will be reached in finite time.

The idea of verifying properties of the above type via discretization of a hybrid system, obtained using conservative approximations, is by no means new, see e.g. [51], [28], [27] and [11]. What characterizes our approach is that we produce two different discrete approximations, one describing an upper bound on allowed (accepted) behavior, and another describing a lower bound on guaranteed (accepted and not excepted) behavior. These discrete systems are formalized in terms of two different finite state machines, an acceptor and an exceptor. In addition, we develop a third discrete device, a reflector, which can be used to obtain tighter bounds on what behavior can be guaranteed.

In Section 9.2 we describe how we derive an abstracted discrete state space and discuss some of its properties. In Section 9.3 we explain why and how we obtain the different transition relations. Section 9.4 deals with the definition of an output function for the FSMs and in Section 9.5 we illustrate the concepts defined via a simple example. Finally, in Section 9.6 we briefly discuss some alternatives for performing the manipulations needed for discretizing the switched system.

9.2 Discrete Plant States

In this section we define the discretization of the state space and discuss the concepts of boundedness and adjacency for the discrete states obtained.

9.2.1 Discrete State Space

The set of plant states is defined by the partition imposed by the generator through the set of hyperplanes. We associate each element of the discrete state vector with the two open halfspaces $\mathcal{H}_i^- = \{x \in \mathbb{R}^n \mid c_i^T x - d_i < 0\}$ and $\mathcal{H}_i^+ = \{x \in \mathbb{R}^n \mid c_i^T x - d_i > 0\}$, as well as the hyperplane $\mathcal{H}_i$ separating them, according to

$$\begin{aligned} x \in \mathcal{H}_i^- &\Leftrightarrow \tilde{x}_i = -1 \\ x \in \mathcal{H}_i &\Leftrightarrow \tilde{x}_i = 0 \\ x \in \mathcal{H}_i^+ &\Leftrightarrow \tilde{x}_i = 1 \end{aligned} \tag{9.1}$$

where $1 \leq i \leq r$ and r is the number of hyperplanes. These expressions can be split into two parts. First, we have a discretization relation defined by a formula $D_i : \mathbb{R}^n \times \{-1, 0, 1\} \to \mathbb{B}$ associating a discrete variable with each set,

$$\begin{aligned} D_i(x, \tilde{x}_i) = \big[&[H_i^-(x) \to \tilde{x}_i = -1]\wedge \\ &[H_i(x) \to \tilde{x}_i = 0]\wedge \\ &[H_i^+(x) \to \tilde{x}_i = 1]\big] \end{aligned} \tag{9.2}$$

where $1 \leq i \leq r$. Second, we have a reconstruction relation defined by the formula $C_i : \{-1, 0, 1\} \times \mathbb{R}^n \to \mathbb{B}$ associating a set with every discrete variable,

$$\begin{aligned} C_i(\tilde{x}_i, x) = \big[&[\tilde{x}_i = -1 \to H_i^-(x)] \wedge \\ &[\tilde{x}_i = 0 \to H_i(x)] \wedge \\ &[\tilde{x}_i = 1 \to H_i^+(x)]\big] \end{aligned} \tag{9.3}$$

where $1 \leq i \leq r$.

Remark 9.1
Recall (from Section 2.2) that $\tilde{x}_i$ denotes the i-th element of the vector $\tilde{x}$, while $\tilde{x}^j$ refers to a specific vector labeled j.

Let us examine how these relations can be used.

Example 9.1 (Discretization and Reconstruction) *Consider the sets defined by Boolean combinations of the formulas*

$$\begin{aligned} H_1(x) &= [x_1 - 1 = 0] \\ H_2(x) &= [x_2 - 1 = 0] \\ H_3(x) &= [x_1 - x_2 - 4 = 0] \\ H_1^-(x) &= [x_1 - 1 < 0] \\ &\vdots \end{aligned} \tag{9.4}$$

From Figure 9.1 we see that the open set $\mathcal{P}_{\tilde{x}^1} = \{x \in \mathbb{R}^2 \mid P_{\tilde{x}^1}(x)\}$ is obtained as the intersection of the halfspaces defined by the constraints $[x_1 - 1 > 0]$, $[x_2 - 1 > 0]$ and $[x_1 - x_2 - 4 < 0]$. In other words, $H_1^+(x)$, $H_2^+(x)$ and $H_3^-(x)$ are the only constraints which are satisfied for points in $\mathcal{P}_{\tilde{x}^1}$ and we can represent the set using $P_{\tilde{x}^1}(x) = H_1^+(x) \wedge H_2^+(x) \wedge H_3^-(x)$. Using the discretization relation, we find that

$$\begin{aligned} \tilde{x}^1 &= \{\tilde{x} \in \{-1, 0, 1\}^3 \mid \exists (x \in \mathcal{P}_{\tilde{x}^1}) \big[\bigwedge_{1 \leq i \leq 3} D_i(x, \tilde{x}_i) \big] \} \\ &= \{\tilde{x} \in \{-1, 0, 1\}^3 \mid \exists x \big[P_{\tilde{x}^1}(x) \wedge \bigwedge_{1 \leq i \leq 3} D_i(x, \tilde{x}_i) \big] \} \\ &= \{\tilde{x} \in \{-1, 0, 1\}^3 \mid [\tilde{x}_1 = 1] \wedge [\tilde{x}_2 = 1] \wedge [\tilde{x}_3 = -1] \} \\ &= (1, 1, -1)^T \end{aligned} \tag{9.5}$$

Conversely, the set corresponding to the vector $\tilde{x}^2 = (0, -1, -1)^T$ can be obtained using the reconstruction relation. That is

$$\begin{aligned} \mathcal{P}_{\tilde{x}^2} &= \{x \in \mathbb{R}^2 \mid \bigwedge_{1 \leq i \leq 3} C(\tilde{x}_i^2, x)\} \\ &= \{x \in \mathbb{R}^2 \mid H_1(x) \wedge H_2^-(x) \wedge H_3^-(x)\} \\ &= \{x \in \mathbb{R}^2 \mid [x_1 - 1 = 0] \wedge [x_2 - 1 < 0] \wedge [x_1 + x_2 - 4 < 0]\} \\ &= \{x \in \mathbb{R}^2 \mid [x_1 - 1 = 0] \wedge [x_2 - 1 < 0]\} \end{aligned} \tag{9.6}$$

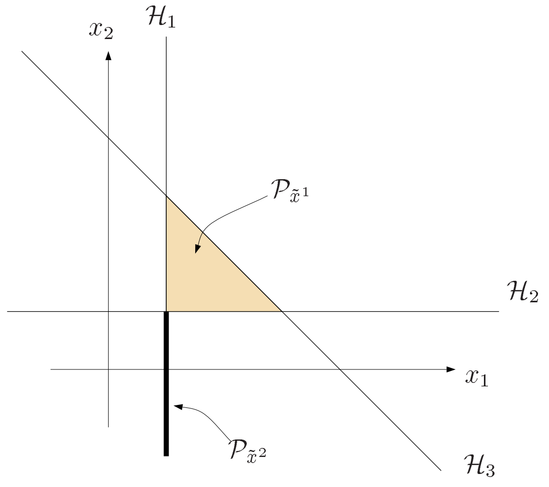


Figure 9.1: Sets in the continuous state space.

where we dropped the constraint $H_3^-(x)$ due to its redundancy. Finally, consider a set corresponding to the vector $\tilde{x}^3 = (-1,-1,1)$. The reconstruction now yields

$$\begin{aligned}\mathcal{P}_{\tilde{x}^3} &= \{x \in \mathbb{R}^2 \mid H_1^-(x) \wedge H_2^-(x) \wedge H_3^+(x)\} \\ &= \{x \in \mathbb{R}^2 \mid [x_1 - 1 < 0] \wedge [x_2 - 1 < 0] \wedge [x_1 + x_2 - 4 > 0]\} \\ &= \emptyset\end{aligned} \tag{9.7}$$

since $\mathcal{P}_{\tilde{x}^3}$ is defined by an inconsistent set of inequalities, i.e., the formula $\exists x \big[[x_1 - 1 < 0] \wedge [x_2 - 1 < 0] \wedge [x_1 + x_2 - 4 > 0]\big]$ has the value `false`. The conclusion is that the vector $\tilde{x}^3 = (-1,-1,1)$ does not describe a valid discrete state.

A valid assignment of values to elements of the vector $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_r)$, say $\tilde{x}'$, thus represents an intersection of hyperplanes and open halfspaces. This intersection is either an open convex polyhedron of dimension $k = n - l$ where $l < n$ is the number of elements in $\tilde{x}'$ which equal zero (cf. $\mathcal{P}_{\tilde{x}^1}$ of Example 9.1 with $k = 2$ and $\mathcal{P}_{\tilde{x}^2}$ where $k = 1$) or, if $l = n$, $\mathcal{P}_{\tilde{x}'}$ degenerates to a single point in the continuous state space. Either way, $\tilde{x}'$ should define a nonempty polyhedron in the continuous state space,

$$\mathcal{P}_{\tilde{x}'} = \{x \in \mathbb{R}^n \mid P(\tilde{x}', x)\} \neq \emptyset \tag{9.8}$$

where $P : \{-1,0,1\}^r \times \mathbb{R}^n \to \mathbb{B}$ is a constraint representing the polyhedron,

$$P(\tilde{x}, x) = \bigwedge_{1 \le i \le r} C_i(\tilde{x}_i, x) \tag{9.9}$$

These valid assignments constitute our discrete state space.

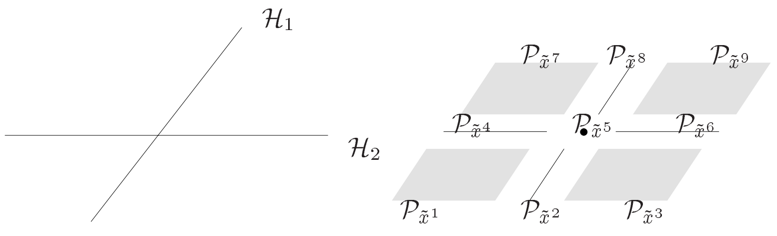
Definition 9.1 (Discrete State Space)
The set of DEDS plant states is defined as the set of states which satisfy the formula $\tilde{X} : \{-1, 0, 1\}^r \to \mathbb{B}$ *obtained as*

$$\tilde{X}(\tilde{x}) = \exists x \big[\bigwedge_{1 \le i \le r} D_i(x, \tilde{x}_i) \big] \tag{9.10}$$

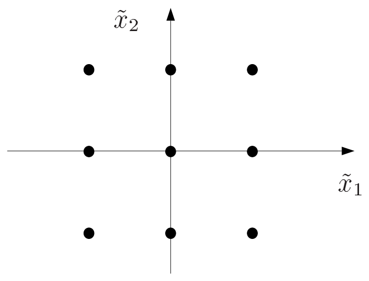
As a result, we obtain a partition of the continuous state space into a collection of open convex polyhedra and their vertices,

$$\mathbb{R}^n = \bigcup_{\tilde{X}(\tilde{x})} \mathcal{P}_{\tilde{x}} = \{x \in \mathbb{R}^n \mid \exists \tilde{x} \big[\tilde{X}(\tilde{x}) \wedge P(\tilde{x}, x)\big]\} \tag{9.11}$$

The partition is illustrated in the case of two hyperplanes in $\mathbb{R}^2$ in Figure 9.2.


$\mathcal{H}_1$
$\mathcal{H}_2$
$\mathcal{P}_{\tilde{x}^7}$
$\mathcal{P}_{\tilde{x}^8}$
$\mathcal{P}_{\tilde{x}^9}$
$\mathcal{P}_{\tilde{x}^4}$
$\mathcal{P}_{\tilde{x}^5}$
$\mathcal{P}_{\tilde{x}^6}$
$\mathcal{P}_{\tilde{x}^1}$
$\mathcal{P}_{\tilde{x}^2}$
$\mathcal{P}_{\tilde{x}^3}$

(a) Continuous state space (b) State space partition



(c) Discrete state space

Figure 9.2: Partitioning the state space.

Remark 9.2
For sakes of clarity, we exclude from our considerations degenerate cases where $l > n$*, i.e., the number of equality constraints exceeds the state space dimension. A*

discussion of degenerate solutions to linear programming problems can be found, for instance, in [4].

Having obtained the discrete states, we examine some of their properties which we will find useful.

9.2.2 Bounded Discrete States

We will need to determine whether a given discrete state $\tilde{x}'$ represents a bounded polyhedron, i.e., whether $\mathcal{P}_{\tilde{x}'}$ is a polytope. An unbounded convex polyhedron $\mathcal{P}_{\tilde{x}'}$, $\tilde{x}' \neq 0$, contains a ray $\{x = x_0 + tv \mid t \geq 0 \ , \ v \in \mathbb{R}^n \ , \ v \neq 0\}$ with base x_0 and direction v, i.e., the formula

$$\exists v \big[[v \neq 0] \wedge [\tilde{x}' \neq 0] \wedge \bigwedge_{1 \leq i \leq r} [\tilde{x}'_i c_i^T v \geq 0]\big] \tag{9.12}$$

holds for such a polyhedron. When we want to obtain discrete states corresponding to bounded polyhedra, we need to ensure that the above condition does not hold.

Definition 9.2 (Bounded Discrete States)
A discrete state is called bounded *if it represents a bounded polyhedron. The set of bounded discrete states is thus defined by*

$$\mathcal{B} = \{\tilde{x} \in \tilde{\mathcal{X}} \mid B(\tilde{x})\} \tag{9.13}$$

where $B : \tilde{\mathcal{X}} \to \mathbb{B}$ is the constraint

$$B(\tilde{x}) = \forall v \big[[v = 0] \vee [\tilde{x}' = 0] \vee \bigvee_{1 \leq i \leq r} [\tilde{x}_i c_i^T v < 0]\big] \tag{9.14}$$

We proceed and define a useful concept involving pairs of discrete states.

9.2.3 Adjacent Discrete States

The concept of adjacency is intuitive when considering geometric objects like polyhedra; two open polyhedra of equal dimension are adjacent if the intersection of their closures is nonempty. In a graph theoretic context the same concept intuitively refers to two nodes connected by an edge and this notion is directly applicable to a graph-like structure as finite state machines. We thus define two discrete states as adjacent if the underlying geometric objects (adjacent or not) allow a transition from one discrete state to the other.

Since we allow only one event to occur at a time, we restrict the transitions to those leading from a polyhedron to one of its facets or vice versa. Corresponding constraints thus hold for the involved discrete states.

Definition 9.3 (Adjacent Discrete States)
Two discrete states $\tilde{x}^1, \tilde{x}^2 \in \mathcal{X}$ are called adjacent *if they correspond to a polyhedron and one of its facets, i.e., if*

$$\overline{\mathcal{P}}_{\tilde{x}^1} \cap \mathcal{P}_{\tilde{x}^2} \neq \emptyset \quad \textit{and} \quad \dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = 1 \tag{9.15}$$

or

$$\mathcal{P}_{\tilde{x}^1} \cap \overline{\mathcal{P}}_{\tilde{x}^2} \neq \emptyset \quad \textit{and} \quad \dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = -1 \tag{9.16}$$

where $\overline{\mathcal{P}}$ denotes the closure of $\mathcal{P}$, $\overline{\mathcal{P}} = \mathcal{P} \cup \partial\mathcal{P}$.

An interesting consequence of the definition of our discrete state space is that adjacent states can be defined in terms of the distance between code words representing the states. This provides an alternative, easily computed, expression for the relation containing adjacent states, namely

$$\tilde{\mathcal{A}} = \{(\tilde{x}, \tilde{y}) \in \tilde{\mathcal{X}} \times \tilde{\mathcal{X}} \mid \tilde{A}(\tilde{x}, \tilde{y})\} \tag{9.17}$$

where $\tilde{A} : \tilde{\mathcal{X}} \times \tilde{\mathcal{X}} \to \mathbb{B}$ is a constraint representing states with unity distance,

$$\tilde{A}(\tilde{x}, \tilde{y}) = [d_c(\tilde{x}, \tilde{y}) = 1] \tag{9.18}$$

For instance, in Figure 9.2 the states $(\tilde{x}^1, \tilde{x}^2)$ and $(\tilde{x}^2, \tilde{x}^5)$ are adjacent, while $(\tilde{x}^1, \tilde{x}^3)$ and $(\tilde{x}^1, \tilde{x}^5)$ are not.

In addition to adjacency, the dynamics of the underlying continuous system must dictate that a transition can take place. We thus need to derive the transition relations for our discrete event systems.

9.3 Transition Relations

In this section we give a motivation of what properties the transition relations should have and give conditions which can be used in order to obtain transition relations satisfying these properties. We then obtain three different kind of state transitions resulting from different conditions, and define corresponding finite state machines having these transitions as their transition relations. Finally, we provide a brief summary of the results obtained.

9.3.1 Abstracting from Switched Dynamics

We start by motivating our approach to discretizing the switched system via conservative approximations, i.e., abstractions. This is done by discussing some simple examples and how different dynamics can be approximated in this way.

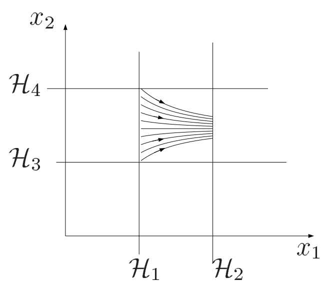
Consider a switched system with a discrete state, $\tilde{x}'$, corresponding to the region

$$\begin{aligned} \mathcal{P}_{\tilde{x}'} &= \{x \in \mathbb{R}^2 \mid H_1^+(x) \wedge H_2^-(x) \wedge H_3^+(x) \wedge H_4^-(x)\} \\ &= \{x \in \mathbb{R}^2 \mid [x_1 - 2 > 0] \wedge [x_1 - 4 < 0] \\ &\qquad \wedge [x_2 - 2 > 0] \wedge [x_2 - 4 < 0]\} \end{aligned} \tag{9.19}$$

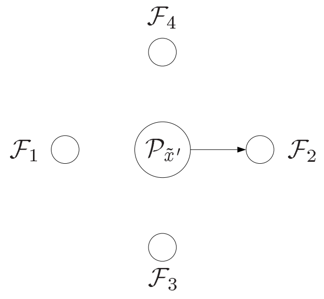
Assume that in this discrete state, the continuous dynamics are given by the affine state equation

$$\dot{x} = \begin{pmatrix} 1/2 & 0 \\ 0 & -1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 3 \end{pmatrix} \tag{9.20}$$

Some trajectories, typical for this mode, are shown in Figure 9.3a. Let us focus



(a) Example dynamics of a discrete mode.



(b) An "equivalent" transition diagram.

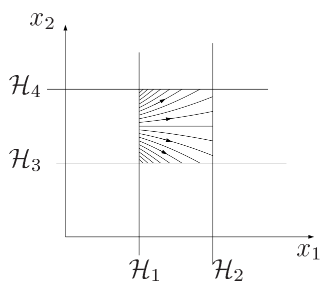
Figure 9.3: Abstracting from switched dynamics - An example.

on transitions from the open region $\mathcal{P}_{\tilde{x}'}$ to one of its facets. For convenience, let's enumerate the facets in accordance with the hyperplane which contains it, i.e., $\mathcal{F}_1 = \{x \in \mathbb{R}^2 \mid H_1(x) \wedge H_2^-(x) \wedge H_3^+(x) \wedge H_4^-(x)\}$, etc. Clearly, the only transition possible in this case is from $\mathcal{P}_{\tilde{x}'}$ to $\mathcal{F}_2$. In addition, we find that for all initial points in $\mathcal{P}_{\tilde{x}'}$, we are bound to exit the region via the same facet. Hence, the dynamics of the region can be seen as equivalent to the transition diagram in Figure 9.3b, provided that we are satisfied with the level of granularity associated with treating $\mathcal{P}_{\tilde{x}'}$ as an equivalence class.

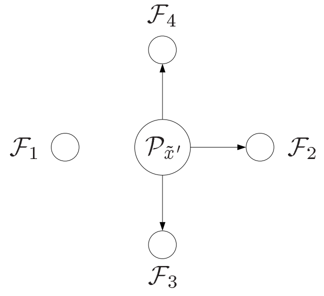
So far, so good. Consider now the case when the continuous dynamics of the discrete mode are given by

$$\dot{x} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ -3 \end{pmatrix} \tag{9.21}$$

This will result in trajectories having the general form shown in Figure 9.4a. Again, we can obtain a transition diagram giving an abstracted view of the transitions of the switched system, this is shown in Figure 9.4b. But now we have to be careful how we interpret the transition diagram. Clearly, any transition shown in the diagram may occur in the switched system. However, there is no way of determining which one of the three possible transitions will occur, knowing only that we start from some point in $\mathcal{P}_{\tilde{x}'}$. All we know for sure is that we will leave $\mathcal{P}_{\tilde{x}'}$ via some of the three facets. Here, we find the concept of non-determinism useful and hence we provide a definition in the context of switched system.



(a) Example dynamics of a discrete mode.



(b) An "equivalent" transition diagram.

Figure 9.4: Abstracting from switched dynamics - An example.

Definition 9.4 (Non-determinism)
We say that a transition to a facet is guaranteed to occur non-deterministically, provided that the corresponding facet belongs to the set of facets where a transition is guaranteed to occur to some arbitrary facet in the set.

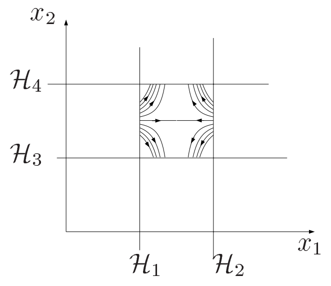
Proceeding with illustrative examples, consider now a case where the mode dynamics are are governed by

$$\dot{x} = \begin{pmatrix} -1/2 & 0 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} 3/2 \\ -3 \end{pmatrix} \tag{9.22}$$

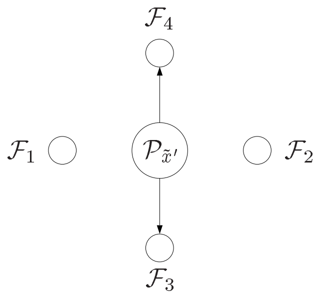
The behavior in this mode is shown in Figure 9.5a. In contrast to the two previous examples, we do not have coinciding transition diagrams for possible and guaranteed transitions. In fact, no transitions can be guaranteed at the discrete level of abstraction, since $\mathcal{P}_{\tilde{x}'}$ contains a stationary point (at $x = (2,2)^T$). Hence, we need two distinct finite state machines to describe the behavior of the system, one which describes possible transitions, and another which describes guaranteed transitions. These are shown in Figure 9.5b and Figure 9.5c, respectively.

Now, if we were to exclude hysteresis conditions from our framework, the only way transitions from a (bounded) region could be prevented would be the existence of stationary points. Since we find that allowing hysteresis in our models gives us valuable modeling power, we need to deal with slightly more complicated situations. Consider, for instance, the case where we switch between two different continuous dynamics, corresponding to two different controller states, within the same region. That is, we define the plant using

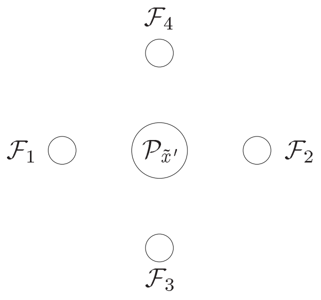
$$\dot{x} = \begin{pmatrix} 1/2 & 0 \\ 0 & -1 \end{pmatrix} x + \begin{pmatrix} -10 + 26u \\ 1 + 4u \end{pmatrix} \tag{9.23}$$



(a) Example dynamics of a discrete mode.



(b) A transition diagram showing possible transitions.



(c) A transition diagram showing guaranteed transitions.

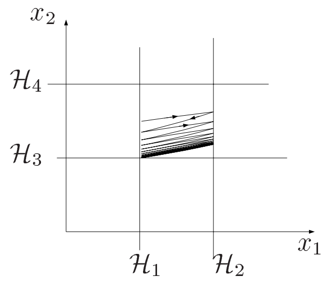
Figure 9.5: Abstracting from switched dynamics - An example.

where events associated with the hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ cause the controller to change its state so that hitting $\mathcal{H}_1$ gives $u = 1$ and entering $\mathcal{H}_2$ results in $u = 0$. This situation is shown in Figure 9.6a. Again, we get different state machines, shown in Figures 9.6b and 9.6c, expressing different properties of the switched system. And due to the possibility of a limit cycle where the controller toggles between its internal states, we are not able to guarantee that any transitions will be made from the region $\mathcal{P}_{\tilde{x}'}$.

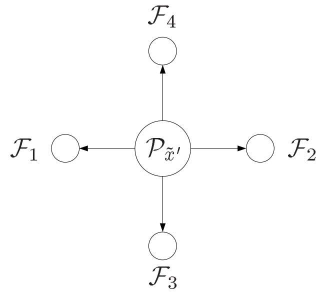
Finally, we consider another case involving discrete controller dynamics. But now, the dynamics are given by

$$\dot{x} = \begin{pmatrix} 1/2 & 0 \\ 0 & -1 \end{pmatrix} x + \begin{pmatrix} -10 + 26u \\ 5 \end{pmatrix} \tag{9.24}$$

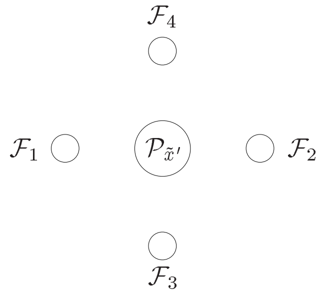
The result is shown in Figure 9.7a. The simulation suggests that, due to the fact that both continuous dynamics drive the state trajectory towards the facet $\mathcal{F}_4$, we



(a) Example dynamics of a discrete mode.



(b) A transition diagram showing possible transitions.



(c) A transition diagram showing guaranteed transitions.

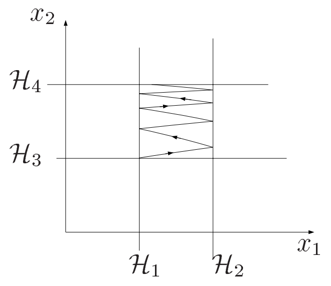
Figure 9.6: Abstracting from switched dynamics - An example.

should be able to conclude that a transition to that facet will eventually be made. We may thus overlook the intermediate transitions to $\mathcal{F}_1$ and $\mathcal{F}_2$ since their only overall effect is to constrain the trajectory within $\mathcal{P}_{\tilde{x}'}$ until $\mathcal{F}_4$ is reached. This is reflected in the finite state machines representing possible and guaranteed behavior, and shown in Figure 9.7b and Figure 9.7c, respectively.

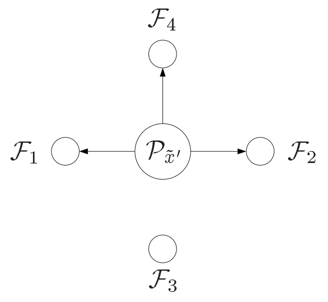
Summarizing this discussion, we make some key observations about transitions from a polyhedron.

- All transitions where the vector field at a facet makes an acute angle with the facet outward normal may occur. We say that these transitions are accepted since they might be admitted by the switched system.
- All accepted transitions will occur (non-deterministically) within a finite time interval, except

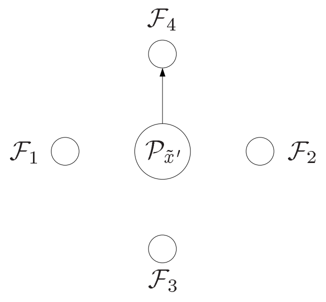
   a) if the polyhedron is not bounded, or



(a) Example dynamics of a discrete mode.



(b) A transition diagram showing possible transitions.



(c) A transition diagram showing guaranteed transitions.

Figure 9.7: Abstracting from switched dynamics - An example.

b) if there is a stationary point in the closure of the polyhedron, or

c) if there is a limit cycle in the closure of the polyhedron.

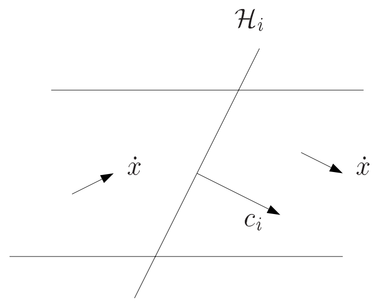
We say that transitions for which a), b) or c) holds are excepted since we cannot guarantee that they will occur in the switched system.

- Accepted transitions which, due to hysteresis, cause the vector field at all points at a facet to change direction from outward to inward, effectively block transitions from the polyhedron closure. As a result the trajectories are pushed into the region again. We say that such transitions are reflected since they cause the trajectories to bounce of the facets.

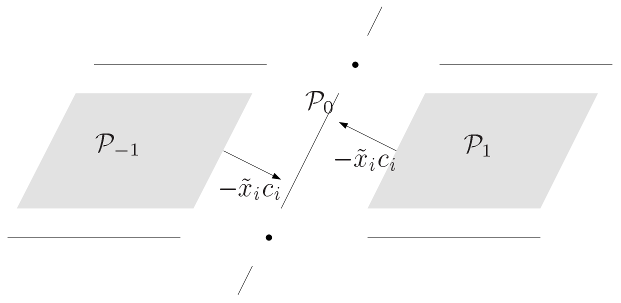
These three observations will play a central role in our approach to discretizing the dynamics of the switched system. And though they have been illustrated here using simple two-dimensional examples, we shall see that these properties can be exploited for any system that fits into our framework. These aspects are formalized in the following sections.

9.3.2 Conditions on Vector Fields

Common to the transition relations obtained from the switched dynamics, is that they are derived using conditions on the vector field in each mode of the underlying switched system, SwS. Let us examine the settings for these conditions when considering transitions involving the hyperplane $\mathcal{H}_i$ in Figure 9.8a. More specifically,



(a) Flow through a hyperplane



(b) Outward normal of $\mathcal{P}_{-1}$ and $\mathcal{P}_1$

Figure 9.8: The settings for transitions through a hyperplane.

we consider transitions involving the polyhedra $\mathcal{P}_{-1}, \mathcal{P}_0$ and $\mathcal{P}_1$ in Figure 9.8b. The transition conditions are expressed in terms of conditions on the angle which the vector field makes with the outward normal of the polyhedra $\mathcal{P}_{-1}$ and $\mathcal{P}_1$. As shown in the figure, this vector is obtained as $-\tilde{x}_i c_i$ where c_i is the normal of the hyperplane $\mathcal{H}_i$. The angle conditions differ for our different types of transitions and for each type of transition we obtain different conditions depending on whether we are considering transitions to or from a facet.

9.3.3 Accepted Transitions

When performing verification, we may want to make sure that a given controlled switched system satisfies certain "safety" properties, that is, the verification procedure involves avoiding certain regions in the continuous state space. We shall see that the concept of accepted transitions provides us with means of doing this, since they give an outer approximation of all transitions possible in the switched system. First, let us formalize what we mean by an accepted transition.

Definition 9.5 (Accepted Transitions)
A transition from a discrete state $\tilde{x}^1 \in \mathcal{X}$ to a discrete state $\tilde{x}^2 \in \mathcal{X}$ is accepted if it may occur in the switched system.

We have stated previously that accepted transitions can be obtained using conditions on the vector field on the boundary of each region. Actually, these conditions differ depending on whether we are considering transition to a polyhedron of upper or lower dimension. We therefore need to consider each case separately.

We start by obtaining the conditions for an accepted transition to a lower dimensional polyhedron.

Proposition 9.1
A necessary condition for a transition from $\mathcal{P}_{\tilde{x}^1}$ to $\mathcal{P}_{\tilde{x}^2}$, where $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ are adjacent and $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = 1$, is that there exists a control symbol and a point on the facet such that the vector field makes an acute or right angle with the facet normal. That is,

$$\exists \tilde{u} \in \tilde{\mathcal{U}}\ ,\ \exists x \in \mathcal{P}_{\tilde{x}^2}\ s.t.\ c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^1 \leq 0 \tag{9.25}$$

where $\tilde{x}_i^2 \neq \tilde{x}_i^1$.

Proof Consider a trajectory, $x(t)$, such that at time t, $x(t) \in \mathcal{P}_{\tilde{x}^2}$ and $\dot{x}(t) = A(\tilde{u}^1)x(t) + b(\tilde{u}^1)$ for some adjacent states $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ such that $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = 1$ and for a control symbol $\tilde{u}^1 \in \tilde{\mathcal{U}}$. We then know that $x(t) \in \overline{\mathcal{P}}_{\tilde{x}^1}$. Assume that our condition is violated and hence $\dot{x}(t)$ is directed towards $\mathcal{P}_{\tilde{x}^1}$. We then know that $x(t^-) \notin \mathcal{P}_{\tilde{x}^1}$. □

An analogous condition can be obtained for transitions to a higher dimensional polyhedron.

Proposition 9.2
A necessary condition for a transition from $\mathcal{P}_{\tilde{x}^1}$ to $\mathcal{P}_{\tilde{x}^2}$, where $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ are adjacent and $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = -1$, is that there exists a control symbol and a point on the facet such that the vector field makes an acute angle with the facet normal. That is,

$$\exists \tilde{u} \in \tilde{\mathcal{U}}\ ,\ \exists x \in \mathcal{P}_{\tilde{x}^1}\ s.t.\ c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^2 > 0 \tag{9.26}$$

where $\tilde{x}_i^1 \neq \tilde{x}_i^2$.

Proof Consider a trajectory, $x(t)$, such that at time t, $x(t) \in \mathcal{P}_{\tilde{x}^1}$ and $\dot{x}(t) = A(\tilde{u}^1)x(t) + b(\tilde{u}^1)$ for some adjacent states $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ such that $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = -1$ and for a control symbol $\tilde{u}^1 \in \tilde{\mathcal{U}}$. We then know that $x(t) \in \overline{\mathcal{P}}_{\tilde{x}^2}$. Assume that our condition is violated and hence $\dot{x}(t)$ is not directed towards $\mathcal{P}_{\tilde{x}^2}$. We then know that $x(t^+) \notin \mathcal{P}_{\tilde{x}^2}$. □

Note that the accepted transitions obtained at the discrete level are indeed an approximation of the transitions that will occur in the switched model. This is due to the fact that the transitions are obtained using criteria on the vector field at the boundaries, whereas the trajectories may never reach the boundaries in question. However, this detailed information regarding the behavior within the region is disregarded by our abstracted discrete model, and we can only hope that our approximation does not introduce so severe artifacts that we fail in performing our verification task.

9.3.4 Acceptor

We now take the concept of accepted transitions a step further and define a finite state machine with a discrete state space as defined in Section 9.2 and a relation containing all accepted transitions as its transition relation.

Definition 9.6 (Acceptor)
Consider a model, SwS, of a switched system as defined in Section 5.2. A discrete event dynamic system which describes the behavior which can occur SwS is called an acceptor *and represented by a finite state machine where all transitions in the FSM may occur in SwS.*

Hence, if a transition is not in the acceptor, we can guarantee that it will not occur in the switched system.

Since the acceptor transition relation should include all possible transitions in the underlying switched system, and since all transitions of the switched system must satisfy either Proposition 9.1 or 9.2, it suffices to define possible transitions as those which satisfy the propositions. The following relation is then obtained for acceptor transitions to lower dimensional polyhedra.

Definition 9.7
The relation of acceptor transitions from polyhedra to their facets is represented by $f_a^l : \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \to \mathbb{B}$ defined as

$$f_a^l(\tilde{x}, \tilde{u}, \tilde{x}^+) = \exists x \big[\tilde{A}(\tilde{x}, \tilde{x}^+) \wedge P(\tilde{x}^+, x) \wedge \bigwedge_{1 \le i \le r} [\tilde{x}_i \neq \tilde{x}_i^+ \to c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i \le 0]\big] \tag{9.27}$$

Similarly, we can define conditions for acceptor transitions to states representing higher dimensional polyhedra.

Definition 9.8
The relation of acceptor transitions from facets to polyhedra is represented by $f_a^h : \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \to \mathbb{B}$ *defined as*

$$f_a^h(\tilde{x}, \tilde{u}, \tilde{x}^+) = \exists x \big[\tilde{A}(\tilde{x}, \tilde{x}^+) \wedge P(\tilde{x}, x) \wedge \bigwedge_{1 \le i \le r} [\tilde{x}_i \neq \tilde{x}_i^+ \to c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^+ > 0]\big] \tag{9.28}$$

We are now in a position to define the transition relation for our discrete model.

Definition 9.9 (Acceptor Transition Relation)
The acceptor transition relation is defined as

$$f_a = \{(\tilde{x}, \tilde{u}, \tilde{x}^+) \in \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \mid f_a^l(\tilde{x}, \tilde{u}, \tilde{x}^+) \vee f_a^h(\tilde{x}, \tilde{u}, \tilde{x}^+)\} \tag{9.29}$$

We now turn to our second type of transitions.

9.3.5 Excepted Transitions

Our motivation for considering excepted transitions is, as in the case of accepted transitions, their use for verification. The difference is that now we may want to make sure that a given controlled switched system satisfies certain "planning" properties, that is, the verification procedure involves ensuring that certain regions in the continuous state space *will* be reached. Since excepted transitions provide us with an outer approximation of all transitions which *cannot* be guaranteed to occur (non-deterministically) in the switched system, they will become useful in this context.

Definition 9.10 (Excepted Transitions)
A transition from a discrete state $\tilde{x}^1 \in \mathcal{X}$ *to a discrete state* $\tilde{x}^2 \in \mathcal{X}$ *is excepted if it is not guaranteed to occur non-deterministically in the switched system.*

We noted in Section 9.3.1 that all transitions from an unbounded region, or from a bounded region containing stationary points or limit cycles in its closure, are excepted, that is cannot be guaranteed. Now, checking for stationary points within a region is an easy task, however, detecting limit cycles is a more subtle one. We can, however, provide sufficient conditions for the absence of both. Again, these conditions are in terms of the vector field, but now evaluated in the closure of the region. Furthermore, we provide two different conditions, one of those is applicable when we only have the possibility of transitions to polyhedra of lower dimension and the other applies to the case of transitions to higher dimensional polyhedra.

Proposition 9.3
A sufficient condition for the state trajectory to exit a polyhedron, $\mathcal{P}_{\tilde{x}^1}$*, is that there exists a bounded polyhedron,* $\mathcal{P}_{\tilde{x}^2}$*, where* $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ *are adjacent,* $\dim \mathcal{P}_{\tilde{x}^1} -$

$\dim \mathcal{P}_{\tilde{x}^2} = 1$ *and for all control symbols and all points in the closure of* $\mathcal{P}_{\tilde{x}^1}$*, the vector field makes an acute angle with the facet normal.That is,*

$$\forall \tilde{u} \in \tilde{\mathcal{U}} \ , \ \forall x \in \overline{\mathcal{P}}_{\tilde{x}^1} \ , \ c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^1 < 0 \tag{9.30}$$

where $\tilde{x}_i^2 \neq \tilde{x}_i^1$.

Proof Consider a trajectory, $x(t)$, such that at time t, $x(t) \in \overline{\mathcal{P}}_{\tilde{x}^1}$ and $\dot{x}(t) = A(\tilde{u})x(t) + b(\tilde{u})$ for some adjacent states $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ such that $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = 1$ and for all $\tilde{u} \in \tilde{\mathcal{U}}$. We construct a scalar measure of the distance from $x(t)$ to the hyperplane containing the facet $\mathcal{P}_{\tilde{x}^2}$, $V(x) = \tilde{x}_i^1(c_i^T x - d_i)$ where the index i is determined by $\tilde{x}_i^2 \neq \tilde{x}_i^1$. Clearly $V(x)$ is zero for points on the hyperplane and positive otherwise. We now examine how this distance varies as time evolves,

$$\dot{V}(x) = \frac{dV(x(t))}{dt} = \tilde{x}_i^1 c_i^T \dot{x} = \tilde{x}_i^1 c_i^T (A(\tilde{u})x + b(\tilde{u})) \tag{9.31}$$

Since our condition guarantees that $\dot{V}(x) \leq -\epsilon < 0$ for all $x \in \mathcal{P}_{\tilde{x}^1}$, $\tilde{u} \in \tilde{U}$, we know that the trajectory will eventually either reach the facet or it will leave the polyhedron via another facet. □

Remark 9.3
Note that we here only demand that the facet is bounded, this is a slightly weaker condition than demanding boundedness of $\mathcal{P}_{\tilde{x}^1}$*, but still a sufficient one.*

Similarly, we obtain a condition for a (deterministic) transition to a higher dimensional polyhedron.

Proposition 9.4
A sufficient condition for the state trajectory to exit a polyhedron $\mathcal{P}_{\tilde{x}^1}$ *is that there exists a polyhedron,* $\mathcal{P}_{\tilde{x}^2}$*, where* $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ *are adjacent and* $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = -1$*, is that for all control symbols and all points in* $\mathcal{P}_{\tilde{x}^1}$*, the vector field makes an acute angle with the facet normal. That is,*

$$\forall \tilde{u} \in \tilde{\mathcal{U}} \ , \ \forall x \in \mathcal{P}_{\tilde{x}^1} \ , \ c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^2 > 0 \tag{9.32}$$

where $\tilde{x}_i^1 \neq \tilde{x}_i^2$.

Proof Consider a trajectory, $x(t)$, such that at time t, $x(t) \in \mathcal{P}_{\tilde{x}^1}$ and $\dot{x}(t) = A(\tilde{u})x(t) + b(\tilde{u})$ for some adjacent states $\tilde{x}^1, \tilde{x}^2 \in \tilde{\mathcal{X}}$ such that $\dim \mathcal{P}_{\tilde{x}^1} - \dim \mathcal{P}_{\tilde{x}^2} = -1$ and for all $\tilde{u} \in \tilde{\mathcal{U}}$. Due to adjacency we know that $x(t) \in \overline{\mathcal{P}}_{\tilde{x}^2}$. If our condition is fulfilled then $\dot{x}(t)$ is directed towards $\mathcal{P}_{\tilde{x}^2}$ for all $x(t)$ and all possible control values. We then know that $x(t^+) \in \mathcal{P}_{\tilde{x}^2}$. □

Note that it of course suffices that one of these conditions is satisfied, for regions of dimension less than n, typically that would be the condition of Proposition 9.4 although the opposite is of course possible and would correspond to a sliding mode. Observe, as well, that again we have approximated the original dynamics; the fact that we cannot guarantee that a trajectory starting within a region will not leave the region in finite time does not necessarily imply that this will not happen. Again, this may lead to artifacts which prevent us from performing verification.

9.3.6 Exceptor

In line with the case of accepted transitions, we now define a finite state machine having a relation containing all excepted transitions as its transition relation.

Definition 9.11 (Exceptor)
Consider a model, SwS, of a switched system as defined in Section 5.2. A discrete event dynamic system which describes the behavior which cannot be guaranteed to occur in SwS is called an exceptor and represented by a finite state machine where all transitions in the FSM cannot be guaranteed to occur in SwS.

Hence, if a transition is in the acceptor and not in the exceptor, we can guarantee that it will occur non-deterministically, i.e., that it belongs to a set of transitions from a state where one transition in the set is guaranteed to occur.

Since we seek a finite state machine representing the transitions which cannot be guaranteed to occur in the switched system, and every such transition does not satisfy the sufficient conditions given in Propositions 9.3 and 9.4, we define its transitions in terms of the negation of the conditions in the propositions. The following relations are then be obtained for exceptor transitions to lower dimensional polyhedra.

Definition 9.12
The relation of exceptor transitions from polyhedra to their facets is represented by the formula $f_e^l : \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \to \mathbb{B}$ defined as

$$\begin{aligned} f_e^l(\tilde{x}, \tilde{u}, \tilde{x}^+) = \neg\tilde{A}(\tilde{x}, \tilde{x}^+) \vee \neg B(\tilde{x}^+) \vee \\ \forall \tilde{x}^+ \Big[\neg\tilde{A}(\tilde{x}, \tilde{x}^+) \vee \exists x \big[\overline{P}(\tilde{x}, x) \wedge \\ \bigwedge_{1 \le i \le r} [\tilde{x}_i \neq \tilde{x}_i^+ \to c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i \geq 0] \big] \Big] \end{aligned} \tag{9.33}$$

Similarly, we define the relation for transitions to higher dimensional polyhedra.

Definition 9.13
The relation of exceptor transitions from facets to polyhedra is represented by $f_e^h : \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \to \mathbb{B}$ defined as

$$\begin{aligned} f_e^h(\tilde{x}, \tilde{u}, \tilde{x}^+) = \neg\tilde{A}(\tilde{x}, \tilde{x}^+) \vee \\ \forall \tilde{x}^+ \Big[\neg\tilde{A}(\tilde{x}, \tilde{x}^+) \vee \exists x \big[P(\tilde{x}, x) \wedge \\ \bigwedge_{1 \le i \le r} [\tilde{x}_i \neq \tilde{x}_i^+ \to c_i^T(A(\tilde{u})x + b(\tilde{u}))\tilde{x}_i^+ \leq 0] \big] \Big] \end{aligned} \tag{9.34}$$

And finally we can define the transition relation for this variant of our discrete model.

Definition 9.14 (Exceptor Transition Relation)
The exceptor transition relation is defined as

$$f_e = \{(\tilde{x}, \tilde{u}, \tilde{x}^+) \in \tilde{\mathcal{X}} \times \tilde{\mathcal{U}} \times \tilde{\mathcal{X}} \mid f_e^l(\tilde{x}, \tilde{u}, \tilde{x}^+) \vee f_e^h(\tilde{x}, \tilde{u}, \tilde{x}^+)\} \tag{9.35}$$

9.3.7 Reflected Transitions

The concept of non-determinism, introduced by necessity when abstracting from the switched model, constitutes one of the main obstacles when performing verification where we want to guarantee certain transitions. We therefore introduce reflected transitions as a way of reducing the degree of non-determinism in our discrete model, relying on the observation that these transitions are in a sense transient, i.e., they do not have any significant effect on the overall behavior.

Definition 9.15 (Reflected Transitions)
A transition from a discrete state $\tilde{x}^1 \in \mathcal{X}$ to a discrete state $\tilde{x}^2 \in \mathcal{X}$ is reflected if it implies that its converse occurs immediately.

Note that reflected transitions will in practice only consist of transitions to lower order polyhedra, since the corresponding events are used to trigger the controller, possibly causing it to change its internal state and hence output, which in its turn leads to a change in dynamics which might push the trajectory into the region again.

9.3.8 Reflector

We now construct a finite state machine with a transition relation consisting of reflected transitions.

Definition 9.16 (Reflector)
Consider a model, SwS, of a switched system as defined in Section 5.2. A discrete event dynamic system which describes the transitions out of regions which are inevitably directed again into the region is called a reflector and is represented by a finite state machine where all transitions in the FSM are cancelled by possible next transitions.

Hence, if a transition is in the reflector, we can exclude it from the set of non-deterministically guaranteed transitions, thus reducing the degree of non-determinism.

Since the occurrence of reflected transitions is a property observed using the discrete model, these transitions can be formed entirely on a discrete level using the transition relation of the acceptor.

Definition 9.17 (Reflector Transition Relation)
The reflector transition relation is defined as

$$f_r(\tilde{x},\tilde{u},\tilde{x}^+) = f_a(\tilde{x},\tilde{u},\tilde{x}^+) \wedge \forall \tilde{u}'[f_a(\tilde{x}^+,\tilde{u}',\tilde{x})] \tag{9.36}$$

This concludes our definitions of discrete transition relations.

9.3.9 Transitions Relations - A Summary

At the risk of becoming tedious, we close this section on transition relations by summarizing our main conclusions about the different finite state machines we have obtained through our various definitions of transitions.

- If a transition is not in the acceptor, we can guarantee that it will not occur in the switched system.
- If a transition is in the acceptor and not in the exceptor, we can guarantee that it will occur non-deterministically, i.e., that it belongs to a set of transitions from a state where one transition in the set is guaranteed to occur.
- If a transition is in the reflector, we can exclude it from the set of non-deterministically guaranteed transitions, thus reducing the degree of non-determinism.

The above statements will be the basis for verification of the switched system using discrete approximations.

We will, in Section 9.5, provide a simple example of how the discretization is performed using the different discrete models. Let us, though, first define the output function for the FSMs obtained in this section.

9.4 Output

Finally, we define the output function. It is identical for the different types of approximations and simply assigns the value `true` to the element of the output vector $\tilde{e}$ corresponding to the hyperplane being entered.

Definition 9.18
The output function for states satisfying a transition condition is a mapping $h : \tilde{\mathcal{X}} \times \tilde{\mathcal{X}} \to \mathcal{E}$ such that

$$h(\tilde{x},\tilde{x}^+) = \{\tilde{e} \in \mathcal{E} \mid \bigwedge_{1\leq i\leq r} [\neg[\tilde{x}_i = 0] \wedge [\tilde{x}_i^+ = 0] \leftrightarrow \tilde{e}_1]\} \tag{9.37}$$

Note, however, that we have violated the formal definition of the output of a finite state machine, which should be a function of current state and input. This is of no significance except for the case where we have non-deterministic transitions. Our definition then corresponds to the fact that even if we do not know which one of a set of transitions will be made for a given input, we will be able to distinguish between different paths.

9.5 A Simple Example

Let us now apply these ideas to our simple tank example from Section 5.2.6.

Example 9.2 (Water Tank, contd.) *We start by obtaining the discrete states of the water tank. The generator, consisting of two distinct points, obviously partitions the real line into five "regions", i.e., three open intervals and two points. This can be also be deducted using the formula for the discrete state space*

$$\begin{aligned} X(\tilde{x}) &= \exists x \big[\bigwedge_{1 \le i \le 2} D_i(x, \tilde{x}_i) \big] \\ &= \exists x \Big[\big[[x - x_l < 0] \to [\tilde{x}_1 = -1]\big] \wedge \\ &\qquad \big[[x - x_l = 0] \to [\tilde{x}_1 = 0]\big] \wedge \\ &\qquad \big[[x - x_l > 0] \to [\tilde{x}_1 = 1]\big] \wedge \\ &\qquad \big[[x - x_h < 0] \to [\tilde{x}_2 = -1]\big] \wedge \\ &\qquad \big[[x - x_h = 0] \to [\tilde{x}_2 = 0]\big] \wedge \\ &\qquad \big[[x - x_h > 0] \to [\tilde{x}_2 = 1]\big] \Big] \end{aligned} \tag{9.38}$$

which yields the set of discrete states

$$\begin{aligned} \mathcal{X} &= \{\tilde{x} \in \{-1, 0, 1\}^2 \mid X(x)\} \\ &= \{(-1,-1)^T, (0,-1)^T, (1,-1)^T, (1,0)^T, (1,1)^T\} \end{aligned} \tag{9.39}$$

which we denote $\{\tilde{x}^1, \dots, \tilde{x}^5\}$. The discretization is depicted in Figure 9.9. Furthermore, we may select as an initial state, for instance, the state corresponding to low level, i.e., $\tilde{x}^0 = \tilde{x}^1 = (-1,-1)^T$.
Having obtained the discrete states, we now need to determine what transitions between the states are possible. And of course we are interested in examining how our different types of finite state machines appear in this example.
Starting with the acceptor, we see that without restrictions on the control symbol, $\tilde{u}$, we can always select $\tilde{u}$ so that a transition is possible between adjacent discrete states. However, taking into account the controller implementation in Example 5.5, we see that transitions from $\tilde{x}^2$ to $\tilde{x}^1$ and from $\tilde{x}^4$ to $\tilde{x}^5$ are no longer accepted. The resulting state machine is shown in Figure 9.10. where the transitions admitted by the controller and the states reachable under control are emphasized.
Turning our attention towards the exceptor, we note that without restrictions on control, we may always have a stationary point within any interval, the result being that all transitions from the corresponding state become exceptor transitions. Similarly, we may choose the control signal in a way which prevents us from penetrating, for instance, the point $x = x_l$ from the state $\tilde{x}^1$. However, provided that $x_l > 0$, $-\frac{b}{a} > x_h$ and using the controller of Example 5.5, we will not have a

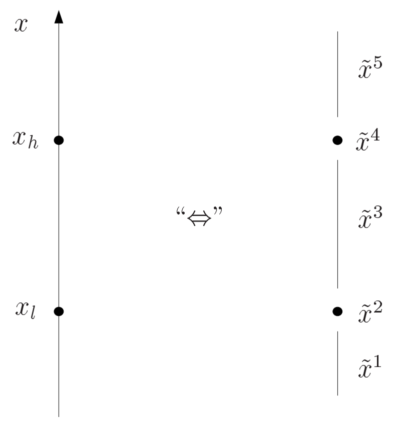


Figure 9.9: Defining discrete states for the tank.

stationary point in any discrete state. Furthermore, the only state with excepted transitions becomes $\tilde{x}^3$; as a consequence we cannot guarantee any transitions out of the corresponding interval. The transitions which we can guarantee are the ones shown in Figure 9.11, since these transitions are accepted and not excepted.
In this example, the reflector is of no interest; we do have two reflected transitions when using the controller (from $\tilde{x}^3$) but since we have no use for means of reducing non-determinism we do not need the reflector. Its use will, however, be exemplified when considering a slightly larger example in Section 11.1.
As a final point of illustration, we show in Figure 9.12 the evolution of the discrete event system corresponding to the actual dynamics of the switched system. Recalling that the acceptor transitions should be a superset of the actual transitions and guaranteed transitions a subset of actual transitions, we see that this, indeed, is the case.

Remark 9.4
Note that in the example we obtained a discrete model of the tank containing five states, while in Section 5.1.1 where we started off with a discrete model, it was sufficient to use three states in order to capture the behavior. However, we see that using the three state model we have no means of guaranteeing that the "low" or "high" states are not reached for a specific controller implementation, due to the lack of intermediate states when making transitions from "normal" level. Hence, our five state discretized model, although more complex, is more suitable for verification purpose.

The example shows that as the plant makes a transition, it outputs events which in their turn force the controller to change its state. In contrast, the controller output, associated only with its state, acts as enabling mechanism for plant transitions.

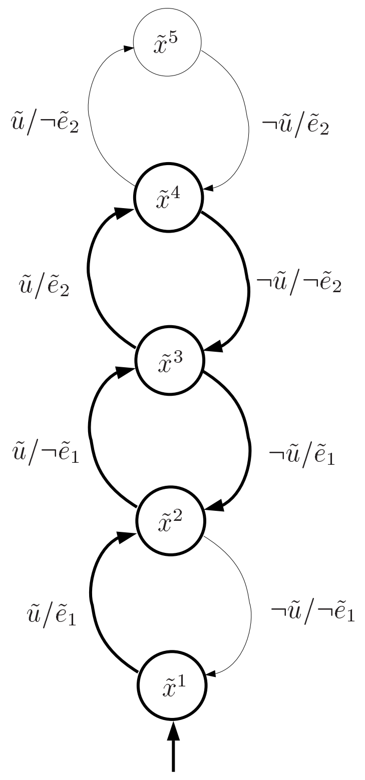

$\tilde{x}^5$
$\tilde{u}/\neg\tilde{e}_2$
$\neg\tilde{u}/\tilde{e}_2$
$\tilde{x}^4$
$\tilde{u}/\tilde{e}_2$
$\neg\tilde{u}/\neg\tilde{e}_2$
$\tilde{x}^3$
$\tilde{u}/\neg\tilde{e}_1$
$\neg\tilde{u}/\tilde{e}_1$
$\tilde{x}^2$
$\tilde{u}/\tilde{e}_1$
$\neg\tilde{u}/\neg\tilde{e}_1$
$\tilde{x}^1$

Figure 9.10: Acceptor FSM for the tank. Thin transitions correspond to "open loop" operation, thick transitions to a controlled tank.

This distinction between passive and active variables resembles the condition/event framework of [56], [36] and provides a notion of causality.

Note that we have not said anything so far about the actual implementation and the computations involved in obtaining the discrete models. And the example treated is sufficiently small to allow even hand computations of the expressions involved. However, it is of course of vital importance that the methods scale up to larger systems, and that problem sizes corresponding to real life applications can be handled. The next section deals with this issue.

9.6 Implementation

Throughout this chapter, we have defined components of a discretized model using expressions which involve variables over both finite and infinite domains, e.g., in our definition of a discrete state space (9.10) and when defining the acceptor and

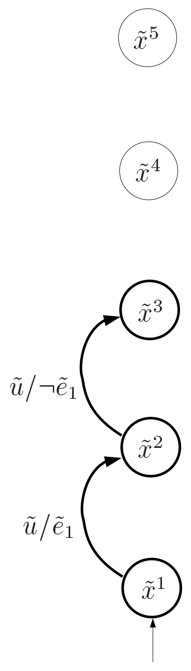


Figure 9.11: FSM expressing guaranteed behavior (accepted and not excepted) of the tank when operated by a controller.

exceptor transition relations (9.29) and (9.35). A question which then may arise is how do we evaluate and work with these expressions?

Let us take as an example the case where we wish to obtain the set of possible discrete states. According to our definition, this is the set of discrete vectors which define a nonempty polyhedron via the reconstruction relation (9.8). Or, equivalently, the set of discrete vectors such that there exists a point in $\mathbb{R}^n$ satisfying the discretization relation (9.10).

A straightforward way to perform this operation would then be to evaluate the reconstruction relation for every possible assignment to the discrete state vector (all $3^{|\tilde{x}|}$ of them). Each assignment results in a set of hyperplanes and halfspaces and we can use a linear programming solver to check if this set has a feasible solution.

An alternative method is applying quantifier elimination, [63], to the formula defining the discrete state space (9.10). Quantifier elimination is a method for transforming a logical formula involving the quantifiers "there exists" ($\exists$) and "for all" ($\forall$) to an equivalent formula without the quantified variables. Since $x \in \mathbb{R}^n$ is quantified in (9.10), quantifier elimination yields an equivalent expression in terms of $\tilde{x} \in \{-1, 0, 1\}$ only. The evaluations of $\tilde{x}$ which satisfy this formula are exactly those which result in a nonempty polyhedron.

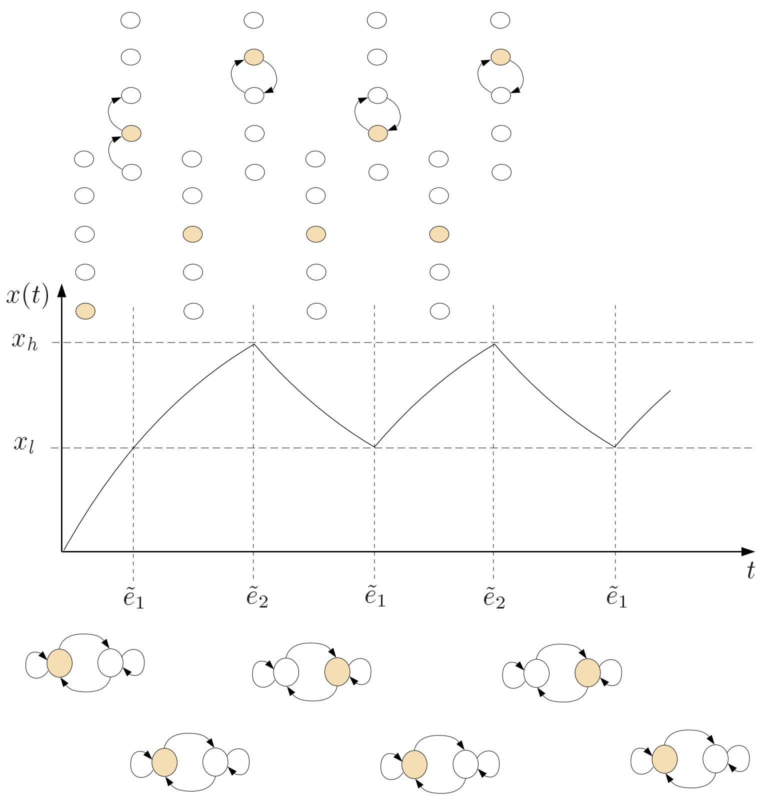


Figure 9.12: Visualization of tank operation. Together with the level as a function of time and the controller state, we show how a FSM counterpart of the tank behaves.

We have mainly used the tool REDLOG , [18], [19], to perform quantifier elimination since it focuses on expressions involving linear and quadratic relations in the real variables. This restriction allows problems with a larger number of variables to be treated. For performing quantifier elimination involving more general expressions, the package QEPCAD [16] may be used.

The drawback of using quantifier elimination is that it involves quite demanding computations. Furthermore, the packages available for performing quantifier elimination (REDLOG [18], QEPCAD [16]) are rather general purpose and not at all tailored to expressions involving variables from mixed domains. This suggests a potential gain from devising a data structure and corresponding algorithms for dealing with expressions of the type we have in our application, i.e., expressions involving variables from both finite and infinite domains.

Although we have discussed the evaluation of mixed expressions in terms of the state space evaluation, same methods can be applied to all other expressions in this chapter since these are of the same form.

These two methods for evaluating the expressions involved have implications on the choice of verification algorithm; the direct evaluation of discrete state vectors is suitable for a method relying on explicit state enumeration while the possibility of obtaining a symbolic expression of the discrete state space facilitates the use of symbolic model checking.

10

Approximations using Power Cones

10.1 Introduction

In this chapter we describe how we can perform model checking of switched systems using the methods developed for power cones in Chapter 4. We first define the type of systems that can be treated and then show that there always exist invariant power cones for these systems. We then demonstrate in detail the operations involved in the reachability procedure by working through the first step for a simple example. Then we describe more generally the reachability algorithm and the subproblems involved and finally we briefly discuss the attainability procedure.

The result of this chapter is an algorithm for model checking of switched systems that needs no human intervention once the system model and specifications are formulated.

10.2 Affine Systems with One Real Pole

We now restrict our attention to dynamic systems where the right hand side of (7.1) is affine in x, that is has the form

$$\dot{x} = Ax + b \tag{10.1}$$

where $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ are constant matrices. Furthermore, we assume that A is nonsingular and satisfies the condition of the following lemma [32].

Lemma 10.1
Suppose that $A \in \mathbb{R}^{n\times n}$ and that there are vectors $v, w \in \mathbb{R}^n$ and an eigenvalue of A, $\lambda \in \mathbb{R}$, such that v is a right eigenvector of A, i.e., $Av = \lambda v$, and w is a left eigenvector, $w^T A = \lambda w^T$. Then there exists a similarity transformation $T = \begin{pmatrix} w_\perp^T & v \end{pmatrix} \in \mathbb{R}^{n\times n}$ such that

$$T^{-1}AT = \begin{pmatrix} \tilde{A} & 0 \\ 0 & \lambda \end{pmatrix}. \tag{10.2}$$

The rows of $w_\perp$ form an orthonormal basis of the orthogonal complement of w.

Proof See [32]. □

The assumptions on A enable us to apply the affine transformation $z = T^{-1}(x + A^{-1}b$ to rewrite the system (10.1) as

$$\dot{z} = \begin{pmatrix} \tilde{A} & 0 \\ 0 & \lambda \end{pmatrix} z. \tag{10.3}$$

It turns out that there always exists a power cone that is invariant for systems in the above form.

10.3 Invariance of Power Cones

As candidates for invariant sets of systems of the form (10.3), we consider the power cones of Section 4.1,

$$\mathcal{PC} = \{ \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \le x_n^r \ , \ x_n \ge 0 \} \tag{10.4}$$

where $P \in \mathbb{R}^{n-1\times n-1}$, $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and x_n, $r \in \mathbb{R}$.

Power cones can also be viewed as sublevel sets of the function

$$V(x) = \tilde{x}^T P \tilde{x} - x_n^r \tag{10.5}$$

We now need a result that gives a constructive way of guaranteeing the invariance of the above sets as defined in Section 7.4.

Lemma 10.2
Let $\dot{x} = Ax$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ be a linear dynamic system. Assume that A has at least one unique real eigenvalue. Then there exists an invariant power cone.

Proof Denote a real eigenvalue of A by λ. We can assume that $A = \operatorname{diag}(\tilde{A}, \lambda)$ since this can always be obtained by a change of variables according to Lemma 10.1. Consider the power cone described by

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \in \mathbb{R}^n \mid \tilde{x}^T P \tilde{x} \leq x_n^r\} \tag{10.6}$$

where $P = P^T \succ 0$, $\tilde{x} \in \mathbb{R}^{n-1}$ and $x_n \geq 0, r \in \mathbb{R}$. Inequality (7.22) with $V(x) = \tilde{x}^T P \tilde{x} - x_n^r$ gives

$$\begin{pmatrix} 2\tilde{x}^T P & -r x_n^{r-1} \end{pmatrix} \begin{pmatrix} \tilde{A}\tilde{x} \\ \lambda x_n \end{pmatrix} = \tilde{x}^T \left(\tilde{A}^T P + P\tilde{A}\right) \tilde{x} - r\lambda x_n^r < 0 \tag{10.7}$$

where the equality follows by symmetrization. For $\mathcal{PC}$ to be invariant this inequality has to be satisfied on the boundary of $\mathcal{PC}$, i.e., $x_n^r = \tilde{x}^T P \tilde{x}$. With this substitution (10.7) becomes

$$\tilde{x}^T \left(\tilde{A}^T P + P\tilde{A} - r\lambda P\right) \tilde{x} < 0 \tag{10.8}$$

and we get the corresponding LMI in P

$$\tilde{A}^T P + P\tilde{A} - r\lambda P \prec 0 \tag{10.9}$$

This LMI has a positive definite solution if and only if the real parts of the eigenvalues of $\tilde{A}$ are less than $r\lambda$, see [33]. Hence, $\mathcal{PC}$ is invariant provided that $r < \frac{2\,\mathrm{Re}\,[\tilde{\lambda}]}{\lambda}$ where $\tilde{\lambda}$ is the eigenvalue of $\tilde{A}$ with the largest real part. □

Example 10.1 *In Figure 10.1 we show some examples of invariant power cones. Note that in order to be able to guarantee that no trajectories will leave the set, we need to ensure that the initial set is a subset of the power cone. This can be checked using Corollary 4.1. Furthermore, note that since power cones are unbounded sets, they serve as invariant sets for unstable systems as well.*

Now, since power cones are invariant sets for systems fulfilling Lemma 10.2, they can, according to Lemma 7.3, be used to conservatively approximate reachable sets for these systems and a given initial set $\mathcal{X}_0$. Furthermore, given a set of goal states $\mathcal{X}_f$, power cones can be used to conservatively approximate its region of attraction. This is stated in the following lemma.

Lemma 10.3
Consider the system

$$\begin{pmatrix} \dot{\tilde{x}} \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} \tilde{A} & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} \tag{10.10}$$

and let $\mathcal{A}[\mathcal{X}_f]$ be the region of attraction for a set of states $\mathcal{X}_f \subseteq \mathbb{R}^n$. Let $\mathcal{PC}$ be an invariant power cone for (10.10) such that $\mathcal{PC} \setminus \mathcal{X}_f$ consists of two disjoint components, $\mathcal{PC}_0$ and $\mathcal{PC}_\infty$ where $0 \in \mathcal{PC}$. Then $(\mathcal{PC}_0 \cup \mathcal{X}_f) \subseteq \mathcal{A}[\mathcal{X}_f]$ $((\mathcal{PC}_\infty \cup \mathcal{X}_f) \subseteq \mathcal{A}[\mathcal{X}_f])$ if $\lambda > 0$ $(\lambda < 0)$.

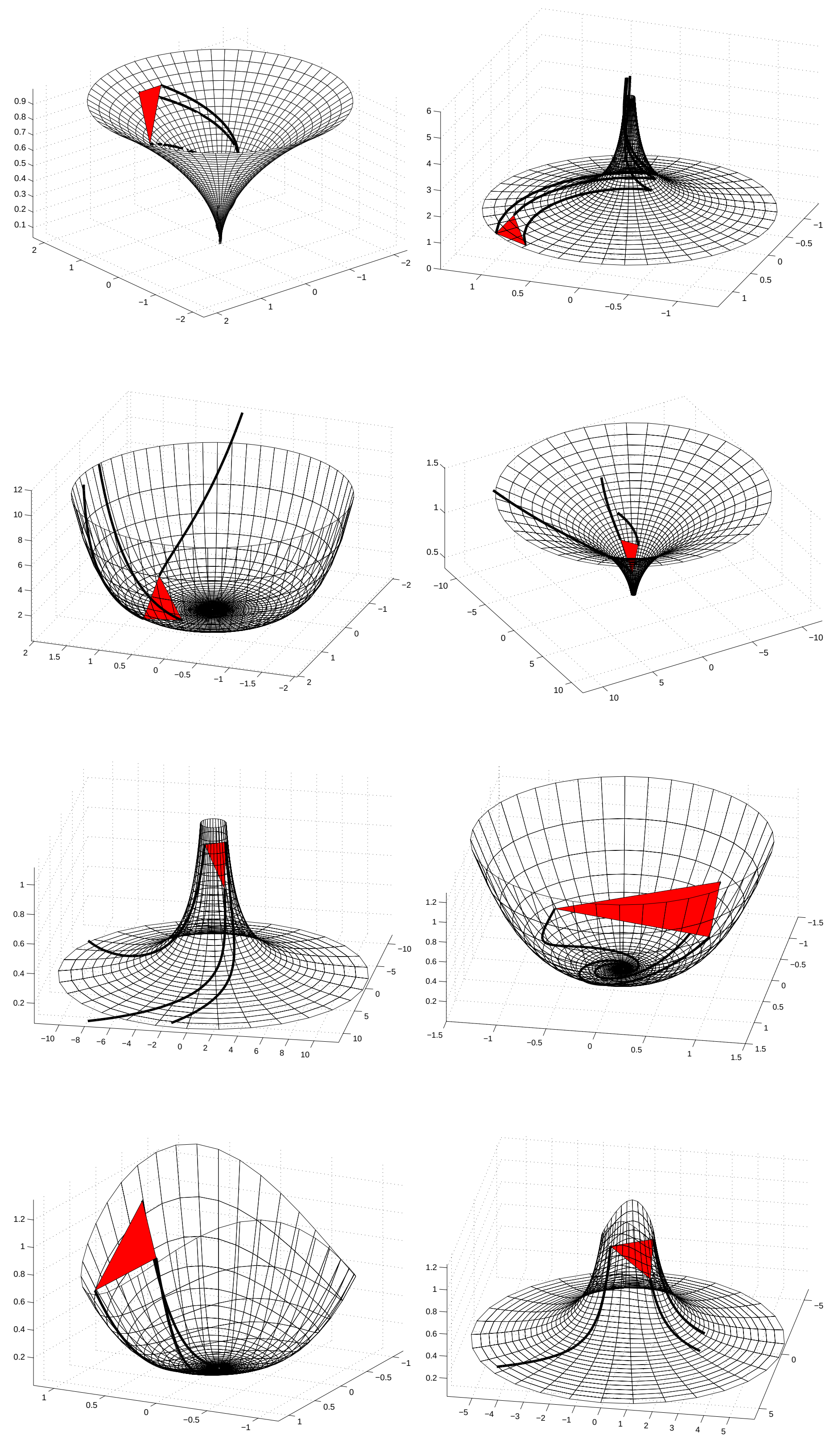


Figure 10.1: Some examples of invariant power cones.

Proof The power cone $\mathcal{PC}$ is invariant and bounded in all directions except the x_n direction. Furthermore, $x_n(t) = e^{\lambda t}x_{n,0}$ is a monotonically increasing (decreasing) function. Let $\mathcal{H}_f = \{(\tilde{x}, x_n)^T \in \mathbb{R}^n \mid x_n = x_{n,f}\}$ be a hyperplane such that $\mathcal{X}_f$ is contained in the open halfspace where $x_n < x_{n,f}$ ($x_n > x_{n,f}$). With $x_0 \in \mathcal{PC}_0$ ($x_0 \in \mathcal{PC}_\infty$) we know that there will exist a timepoint t_f where $x_n(t_f) = x_{n,f}$. This implies that the there has been a time $t < t_f$ where $x(t) \in \mathcal{X}_f$ since $\mathcal{PC}_0 \cup \mathcal{X}_f$ ($\mathcal{PC}_\infty \cup \mathcal{X}_f$) is bounded by $\mathcal{X}_f$.

□

Having developed methods for performing set operations involving power cones and affine and quadratic sets, we are now in a position to use these sets as approximations of reachable sets and regions of attraction, and hence to perform model checking of switched systems using these tools.

10.4 Reachability - an Example

In this section we will work through the details of performing the first step of model checking for a simple switched system. It is worthwhile keeping in mind that although we work out this example by hand, there is nothing in the procedure that requires manual input and hence the procedure is fully automated.

10.4.1 Problem Definition

Consider the switched system shown in Figure 10.2. The system consists of four modes, each corresponding to a fixed value of the discrete controller output $u(t) = u_i$. Hence, we associate with each mode i affine dynamics $\dot{x} = A_i x + b_i, i = 1, \ldots, 4$ and transition surfaces T_{ij} that cause the controller to switch from node i to node j when $x(t)$ hits it.

The mode specific affine dynamics are given by the system matrices and constant vectors

$$A_1 = \begin{pmatrix} -2 & 5 & -4.9 \\ -5 & -2 & 5.1 \\ 0 & 0 & -1.9 \end{pmatrix}, \qquad b_1 = \begin{pmatrix} 7.6 \\ 7.6 \\ 7.6 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} -1.86 & 0.79 & -0.18 \\ -0.79 & -1 & -3.92 \\ 0.18 & 3.92 & -1.04 \end{pmatrix}, \qquad b_2 = \begin{pmatrix} 9.66 \\ 22.78 \\ -11.32 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} -0.7 & -7 & 4.7 \\ 7 & -0.7 & -9.3 \\ 0 & 0 & -3 \end{pmatrix}, \qquad b_3 = \begin{pmatrix} 12.5 \\ -26 \\ 9 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 0.1 & -7 & 4.7 \\ 7 & 0.1 & -9 \\ 0 & 0 & -3 \end{pmatrix}, \qquad b_4 = \begin{pmatrix} 10 \\ -20 \\ 10 \end{pmatrix},$$

and the transition surfaces are given by the hyperplanes

$$\begin{aligned}
\mathcal{T}_{12} &= \{x \in \mathbb{R}^3 \mid x_1 + x_2 + x_3 = 9.6\},\\
\mathcal{T}_{23} &= \{x \in \mathbb{R}^3 \mid x_1 = 6.5\},\\
\mathcal{T}_{32} &= \{x \in \mathbb{R}^3 \mid x_3 = 10\},\\
\mathcal{T}_{34} &= \{x \in \mathbb{R}^3 \mid x_1 = 11\}.
\end{aligned}$$

We let the set of initial states in mode 1 be the convex hull of four points,

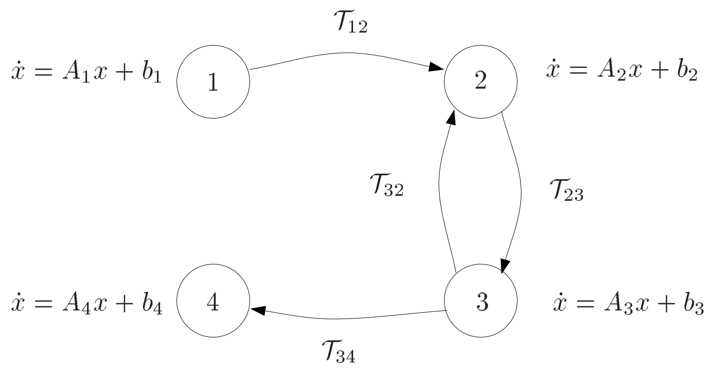


Figure 10.2: A switched system with four modes.

$$\mathcal{X}_{0,1} = \mathbf{Co}\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\-1\\0\end{pmatrix}, \begin{pmatrix}-1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\}$$

and start the model checking by finding an invariant power cone in mode 1 that contains the initial set.

10.4.2 Mode 1

The affine system of mode 1 has a stationary point at $x_{c,1} = -A_1^{-1}b_1 = (4,4,4)^T$. Lemma 10.1 applies, hence we can apply the similarity transformation $z = T_1^{-1}(x - x_{c,1})$ with

$$T_1 = \begin{pmatrix} 0 & 1 & -0.5774\\ -1 & 0 & -0.5774\\ 0 & 0 & -0.5774 \end{pmatrix} \tag{10.11}$$

which gives us the transformed system

$$\dot{z} = \tilde{A}z = \begin{pmatrix} -2 & 5 & 0\\ -5 & -2 & 0\\ 0 & 0 & -1.9 \end{pmatrix} z. \tag{10.12}$$

We need to transform the initial set into these new coordinates as well, this yields

$$\mathcal{X}_0 = \mathbf{Co}\{v_1, v_2, v_3, v_4\} = \mathbf{Co}\{\begin{pmatrix} 0 \\ 1 \\ 6.9282 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 6.9282 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 6.9282 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 6.9282 \end{pmatrix}\}$$

Here, all corner points of the initial polytope have the same z_3 coordinate. An invariant cone containing these points will therefore also contain the polytope due to the convexity of the ellipsoid obtained by intersecting the cone with the hyperplane $\{z \in \mathbb{R}^3 \mid z_3 = 6.9282\}$. Hence, we do not need to apply Corollary 4.1; it suffices to find the smallest invariant power cone containing these points. This leads to the convex optimization problem

$$\begin{array}{ll} \text{minimize} & \log\det P^{-1} \\ \text{subject to} & \tilde{A}^T P + P\tilde{A} - r\lambda P \prec 0, \quad P \succ 0 \\ & \tilde{v}_i^T P \tilde{v}_i \leq -(v_i)_3^r, \quad i = 1, \ldots, 4. \end{array} \tag{10.13}$$

where $r = 2.1053$ and $\lambda = -1.9$ and $\tilde{v}_i$ contains the first two coordinates of each v_i. Solving this problem yields the power cone

$$\mathcal{PC} = \{\begin{pmatrix} \tilde{z} \\ z_3 \end{pmatrix} \in \mathbb{R}^3 \mid \tilde{z}^T \begin{pmatrix} 55.6589 & -0.0000 \\ -0.0000 & 55.6589 \end{pmatrix} \tilde{z} \leq z_3^{2.105}\}. \tag{10.14}$$

A plot of this power cone in z coordinates as well as the initial set and simulation trajectories for the affine system starting at the corners of the polytope are shown in Figure 10.3. Having found the smallest invariant power cone for mode 1, we

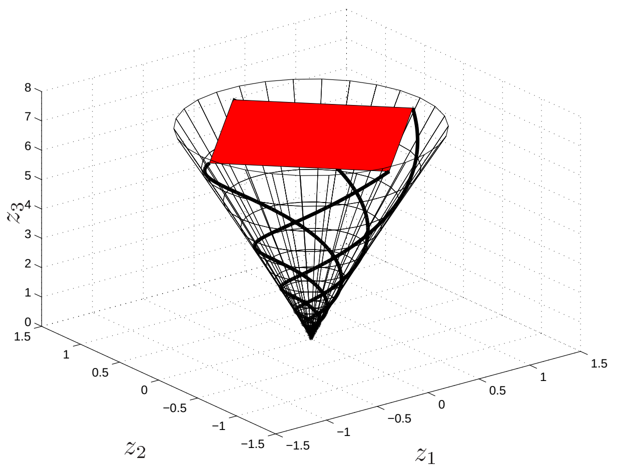


Figure 10.3: The power cone obtained for mode 1 along with the initial set and some simulation trajectories.

need to check if it intersects the switching hyperplane $\mathcal{T}_{12}$ and compute the initial set for mode 2 in case it does.

Checking if the power cone intersects the switching surface can be done automatically by checking if (4.78) has any solutions. Here, it turns out that (4.78) has three solutions, $\underline{z}_n = 1.2475$, $\tilde{z}_n = 1.5605$ and $\overline{z} = 1.8007 \cdot 10^{18}$. Hence, according to Section 4.2.1, the intersection consists of two components, one bounded and one unbounded. The unbounded component of the intersection is far out of our range of interest since no states can be reached outside the interval $[0, 6.9282]$. As a matter of fact, we only need to search for solutions in that interval. Hence, we focus on the former two solutions.

Since $r \in (2, \infty)$, we construct an outer approximation of the intersection using an elliptic cone. This results in an initial set for mode 2,

$$\mathcal{X}_{0,2} = \{z = C^\dagger d + C_{\mathcal{N}} y \mid \hat{y}^T M^T \hat{P}_2 M \hat{y} \leq 0\} \tag{10.15}$$

where

$$\hat{y} = \begin{pmatrix} y \\ 1 \end{pmatrix}, \quad M = \begin{pmatrix} -0.8165 & 0 & 3.2 \\ 0.4082 & -0.7071 & 3.2 \\ 0.4082 & 0.7071 & 3.2 \\ 0 & 0 & 1 \end{pmatrix} \tag{10.16}$$

and

$$\hat{P} = \begin{pmatrix} 48.4797 & -0.0000 & 0 & 0 \\ -0.0000 & 48.4797 & 0 & 0 \\ 0 & 0 & -1 & 0.0696 \\ 0 & 0 & 0.0696 & 0.0048 \end{pmatrix} \tag{10.17}$$

We now need to change the coordinates back to the x coordinate system. A plot of the power cone, along with trajectories and the switching hyperplane $\mathcal{T}_{12}$ is shown in Figure 10.4. We are now in mode 2 with new dynamics and a new initial set described as the intersection of a quadratic and affine set. Hence, we can repeat the procedure for that set.

Let us now take a more general look at the ingredients of the model checking algorithm.

10.5 Reachability - Bad States

Here we describe the procedure for performing reachability analysis. The basic procedure is simple:

1. Compute the smallest invariant power cone containing the initial set.
2. Check if this power cone intersects any of the switching surfaces.
3. Compute the intersection where it exists.

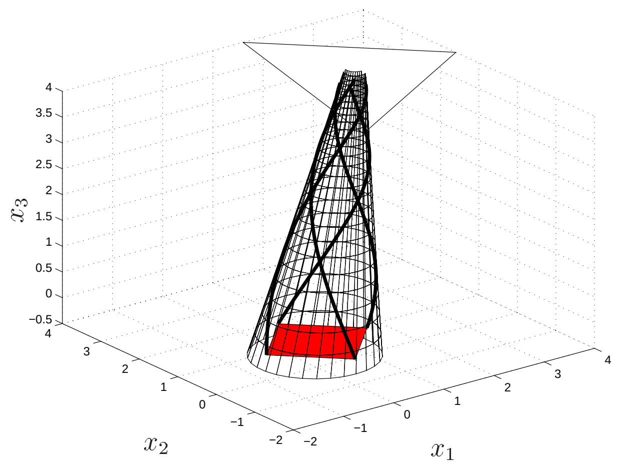


Figure 10.4: The power cone obtained for mode 1 along with simulation trajectories and the switching surface $\mathcal{T}_{12}$.

4. Repeat the process for each of the intersections obtained, using it as an initial set, until a mode is reached that violates the specification or no new reachable sets are found.

Below we discuss in more detail how these steps can be performed automatically using the results from Chapter 4.

A basic problem we need to solve in each step of the reachability procedure is to find the smallest invariant power cone containing the set of initial states for the current mode. That is, we generally need to solve the following constrained optimization problem:

$$\begin{array}{ll} \text{minimize} & \text{size } \mathcal{PC} \\ \text{subject to} & \mathcal{PC} \text{ invariant} \\ & \mathcal{X}_0 \subseteq \mathcal{PC}. \end{array} \tag{10.18}$$

The invariance condition is the LMI of equation (10.9) while the set inclusion condition may be one of those treated in Sections 4.3.1 and 4.3.2. The size minimization is performed by minimizing the function $\log\det P^{-1}$. This problem is convex and can be solved very efficiently.

In addition we need to perform certain transformations and approximations along the way.

The initial set of each mode needs to be transformed according to the mode specific dynamics. And the intersection of the power cone with hyperplanes needs to be transformed back to the original coordinates. For these transformation we use the transformation matrix of Lemma 10.1.

We have to find conservative outer approximations of the intersection of the power cone with the switching hyperplanes. This is done as described in Section 4.2.1 and involves solving the nonlinear single variable equation (4.78). Note that when solving (4.78) in these settings, we only have to look for solutions in the interval $[0, x_{n,max}]$ where $x_{n,max}$ is the largest x_n such that $\mathcal{H}_{x_n} \cap \mathcal{X}_0 \neq \emptyset$.

Finally, we need to be able to formulate the subset relation $\mathcal{X}_0 \subseteq \mathcal{PC}$ of (10.18) in order to solve the problem. The next two sections describe how this is done in the two cases we need to handle, when $\mathcal{X}_0$ is a polytope and when $\mathcal{X}_0$ consists of the intersection between a quadratic and an affine set.

10.5.1 Initial Polytope

In the initial step of the reachability procedure, we typically have a set of initial states given as the continuous region corresponding to a given initial mode m_i. That is, the initial set will be a polyhedron. This polyhedron may not be of full dimension and will then serve directly as an initial set for the next mode, say m_j. In case the initial polyhedron is full dimensional, we will be able to leave it through some of its facets, i.e., an $n-1$ dimensional polytope. In either case, the initial polytope of mode m_j will be of dimension $\leq n-1$. For sake of clarity, we will assume that we are dealing with the $n-1$ dimensional case; the lower dimensional case can be handled with straightforward modifications.

That is, starting in mode m_j we have an initial set which is a polytope

$$\mathcal{X}_0 = \mathcal{P} = \{x \in \mathbb{R}^n \mid c_{ij}^T x = d_{ij} \ , \ c_{ik}^T x \geq d_{ik} \ , \ k \neq j\}. \tag{10.19}$$

We thus seek the smallest invariant power cone containing the polytope $\mathcal{P}$. This leads to the optimization problem

$$\begin{array}{ll} \text{minimize} & \log\det P^{-1} \\ \text{subject to} & \tilde{A}^T P + P\tilde{A} - r\lambda P \prec 0 \ , P \succ 0 \\ & \mathcal{P} \subseteq \mathcal{PC}. \end{array} \tag{10.20}$$

In order to solve () we need to be able to represent the set inclusion $\mathcal{P} \subseteq \mathcal{PC}$. This is solved by first computing the convex hull representation of $\mathcal{P}$, i.e., $\mathcal{P} = \mathbf{Co}\{v_1, \ldots, v_p\}$, and then apply Corollary 4.1.

10.5.2 Initial Quadratic and Affine Set

When the initial set is the intersection of a quadratic and an affine set, $\mathcal{X}_0 = \mathcal{Q} \cap \mathcal{A}$ where

$$\begin{aligned} \mathcal{Q} &= \{x \in \mathbb{R}^n \mid x^T A x + 2b^T x + c \leq 0\} \\ \mathcal{A} &= \{x \in \mathbb{R}^n \mid Cx = d\} \end{aligned} \tag{10.21}$$

we make use of the subset checks developed in Section 4.3.2. That is, we pose the constrained optimization problem

$$\begin{array}{ll} \text{minimize} & \log\det P^{-1} \\ \text{subject to} & \tilde{A}^T P + P\tilde{A} - r\lambda P \prec 0\,, P \succ 0 \\ & M^T\left(\hat{P} - \tau\begin{pmatrix} A & b \\ b^T & c\end{pmatrix}\right)M \preceq 0, \quad \tau \geq 0 \end{array} \tag{10.22}$$

where

$$M = \begin{pmatrix} C_{\mathcal{N}} & C^{\dagger}d \\ 0 & 1 \end{pmatrix} \tag{10.23}$$

and $\hat{P}$ varies depending on the value of the power parameter r according to the following:

- $r \in (0,1)$:

$$\hat{P} = \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad \alpha = 1/a^r \tag{10.24}$$

- $r \in (1,2)$:

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & -1 & x_{n,0} \\ 0 & x_{n,0} & x_{n,0}^2 \end{pmatrix}, \\ x_{n,0} &= \frac{a(r-1)}{r} \quad , \quad \alpha = \frac{1}{ra^{r-1}} \end{aligned} \tag{10.25}$$

- $r \notin [0,2]$:

$$\begin{aligned} \hat{P} &= \begin{pmatrix} \alpha P & 0 & 0 \\ 0 & 0 & -1/2 \\ 0 & -1/2 & x_{n,0} \end{pmatrix}, \\ x_{n,0} &= \frac{a(r-2)}{r} \quad , \quad \alpha = \frac{4}{r^2 a^{r-2}} \end{aligned} \tag{10.26}$$

where a is the larger value of the solutions to equation (4.123) if $r \in (-\infty, 0)$ and the smaller value if $r \in (0, \infty)$.

10.6 Attainability - Goal States

In contrast to the reachability procedure, attainability analysis is performed backwards from the goal states.

1. Find the largest invariant power cone that is divided into two disjoint components by the initial set.
2. Check for intersections with switching surfaces.
3. Use the intersections as final sets in the corresponding modes and add them to the set of initial states.
4. Repeat until either no new modes are found or until the set of initial states is contained in the union of power cones found.

We discuss below the methods from Chapter 4.1 that are used to perform these steps, with the exception of step 1 since we have not devised a method for checking if a set divides a power cone into two disjoint components in this thesis. This is however the only missing link in order to be able to attempt solving the attainability problem using approximations.

The optimization problem that needs to be solved at each step has the form

$$\begin{array}{ll} \text{minimize} & \text{size } \mathcal{PC} \\ \text{subject to} & \mathcal{PC} \text{ invariant} \\ & \mathcal{PC} \setminus \mathcal{X}_f = \mathcal{PC}_0 \cup \mathcal{PC}_\infty, \quad \mathcal{PC}_0 \cap \mathcal{PC}_\infty = \emptyset \end{array} \tag{10.27}$$

Here, the function to be minimized and the invariance condition are the same as for the reachability procedure. However, while the reachability problem has the condition that the initial set is contained in the power cone, we here require that the final set splits the power cone in two disjoint sets. Hence, the final set bounds each half of the power cone and the trajectories will then have to leave the half through the final set.

The same type of transformations as for the reachability procedure need to be performed, i.e., the initial set is transformed using the transformation matrix of Lemma 10.1.

New final sets are approximated using the inner approximations of Section 4.2.2 since we always need to make sure that states will be reached that are guaranteed to lead to the goal states.

10.7 Summary

We have described methods for performing algorithmic model checking given a system model and a specification of either bad states to be avoided or goal states that we wish to attain. The methods are based on the use of invariant power cones and rely on the use of convex optimization as well as the procedures developed in Section 4.1. Once the model and specification are given, no further human intervention is required, the selection of types of invariant cones, as well as approximation techniques is fully automated based on easily checked criteria.

Part IV

Applications

11

Examples and Applications

Here we will consider two different applications of the methods presented in this thesis. By applications we mean examples of sufficient complexity to reflect possible real life situations which may benefit from the tools we provide.

The first example, presented in Section 11.1 is a fictional chemical reaction process, constructed in a way that demonstrates typical challenges when faced with the control of industrial processes. The second example deals with a simple model of the landing gear system of the Swedish JAS 39 Gripen defense and fighter aircraft. This example is treated in Section 11.2.

11.1 Chemical Reactor

In this section, we apply our tools to a model describing a fictional chemical reaction process. Although imaginary, the process resembles many typical process control applications where heuristically derived controllers are designed due to lack of systematic methods for controller design. This makes the process suitable for applying verification to check for errors in the resulting design.

11.1.1 System Description

A schematic figure of the fictional chemical reactor is shown in Figure 11.1. The

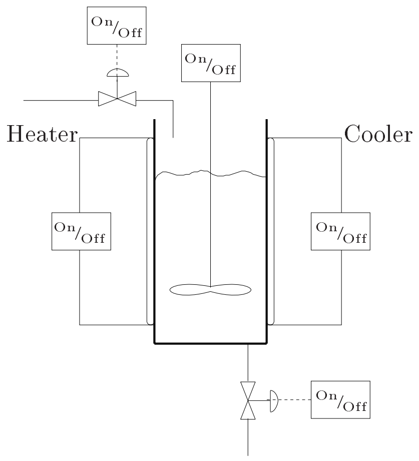


Figure 11.1: A chemical reactor

process consists of a tank which can hold a certain amount of fluid. The fluid level can be changed by opening either an inflow valve or a draining valve extracting the mixture for use in a subsequent process. The valves are considered to be discrete, i.e., they can only be open or closed. We now assume that the fluid flowing into the tank is a mixture of two substances which have the property that they react with each other when the mixture reaches a certain temperature. Furthermore, the chemical reaction that takes place is exothermal, i.e., the reaction produces thermal energy, thus resulting in further heating of the mixture. To regulate the temperature of the fluid we have two devices, a heater and a cooler. Both devices are discrete, that is, they can either be turned on or off. The heater can be used to bring the temperature to a level where the reaction starts and the cooler may be utilized in order to prevent overheating of the system when the reaction is in progress. Finally, we have access to a blender that can be used to mix the fluid and thus aid the transport of heat through the tank.

It should be clear that, due to the relatively large number of discrete actuators affecting the (continuous) state of the system, designing a controller for this process is by no means a trivial task. However, any controller design must strive to achieve some desired behavior, so let us reflect on what behavior is desirable in the above process.

1. The controller should guarantee that the temperature is never below minimum and never above maximum.

2. The controller should guarantee that the tank is never empty and that it never overflows.

3. There shall exist an operating region where the temperature is neither high nor low and the fluid level is neither high nor low. This region should be

invariant, i.e., once entered it should not be left under normal operating conditions.

4. The operating region should always be reached from the initial states in finite time.

We note that properties 1,2 and 3 are typical safety properties, specifying that certain states are never reached and that certain transitions are not made. On the other hand, the fourth property classifies as a goal state property; we wish to guarantee that certain states are reached in finite time.

11.1.2 System Model

We now derive a model of the reactor system, that is we define the plant, controller, actuator and generator. We then use the model to gain further insight into the behavior of the process by simulating some typical trajectories.

Plant

The plant model consist of two continuous signals, states, describing the condition of the system. These are the tank level and the fluid temperature.

The tank level is only influenced by the in- and outflow, the flow in its turn depends on the position of the inflow valve, u_i, and the draining valve, u_d. Using a linearized model of the tank, the flow dynamics can be described with a differential equation with piecewise linear right hand side,

$$\dot{h} = \begin{cases} 0 & \text{if inflow and outflow valves closed} \\ b_h & \text{if inflow valve open and outflow valve closed} \\ -a_h h & \text{if inflow valve closed and outflow valve open} \\ -a_h h + b_h & \text{if inflow and outflow valves open} \end{cases} \tag{11.1}$$

The above model is captured by the single equation

$$\dot{h} = -a_h u_d h + b_h u_i \tag{11.2}$$

where u_d and u_i can take the values 0 (closed) or 1 (open).

The fluid temperature depends of course on the values of the heater, u_h, the cooler, u_c and whether the reaction, represented by an uncontrollable input signal u_r, has started. Furthermore, the value of the blender, u_b influences the heat distribution in the tank and therefore the time constant of the temperature dynamics. These considerations result in a compact differential equation describing the temperature dynamics,

$$\dot{T} = -(a_{T_1}(1 - u_b) + a_{T_2} u_b)T + b_{heat} u_h + b_{cool} u_c + b_{reac} u_r \tag{11.3}$$

where the continuous control signals u_b, u_h, u_c and u_r can take values 0 (off) and 1 (on).

Denoting the level x_1 and the temperature x_2, the system can be described by (5.8) with

$$A(u) = \begin{pmatrix} -a_h u_d & 0 \\ 0 & -(a_{T_1}(1-u_b) + a_{T_2} u_b) \end{pmatrix} \tag{11.4}$$

$$b(u) = \begin{pmatrix} b_h u_i \\ b_{heat} u_h + b_{cool} u_c + b_{reac} u_r \end{pmatrix} \tag{11.5}$$

where the continuous control signals u_* can take values 0 (off) and 1 (on).

Remark 11.1
This application illustrates three nice properties of the modeling formalism. First, the relative ease of modeling using basic physical principles. Second, the modularity of the modeling procedure, that is, we consider how each actuator influences the system and then combine the effects of all actuators at the end. Finally, the resulting model is a very compact description of the resulting, relatively complex ($2^6 = 64$ different state space models), system.

A summary of plant signals is given in Tables 11.1 and 11.2.

Signal	Domain	Interpretation
u_b	$\{0,1\}$	blender signal
u_i	$\{0,1\}$	inflow valve signal
u_d	$\{0,1\}$	draining valve signal
u_h	$\{0,1\}$	heater signal
u_c	$\{0,1\}$	cooler signal
u_r	$\{0,1\}$	reaction signal

Table 11.1: Inputs to the chemical reactor.

Signal	Domain	Interpretation
x_1	$\mathbb{R}$	tank level
x_2	$\mathbb{R}$	fluid temperature

Table 11.2: States (and outputs) of the reactor plant model.

The parameter values used in the model (obtained as “reasonable values” after some physical considerations) are listed in Table 11.3

We have mentioned that the controller design is no easy task. Let us take our best shot at performing that task, enjoying the possibility of being able to verify the design at a later stage.

Parameter	Value ($\cdot 10^{-3}$)
a_h	1.23
a_{T_1}	0.15
a_{T_2}	0.22
b_h	9838
b_{heat}	29.43
b_{cool}	-44.15
b_{reac}	44.15

Table 11.3: Parameter values for the chemical reactor.

Controller

The controller has the discrete output $\tilde{u} = (\tilde{u}_b, \tilde{u}_i, \tilde{u}_d, \tilde{u}_h, \tilde{u}_c, \tilde{u}_r)^T$ (corresponding to the signals just defined) and its design is obtained using the following heuristics:

- Blender, $\tilde{u}_b$: The blender is off when the fluid level is very low and on otherwise.
- Inflow, $\tilde{u}_i$: The inflow valve is open while the fluid level is not high, then it is closed. It stays closed while the fluid level is not low.
- Draining, $\tilde{u}_d$: The drain valve is closed when there is no reaction and open otherwise.
- Heater, $\tilde{u}_h$: The heater is on when there is no reaction and off otherwise.
- Cooler, $\tilde{u}_c$: The cooler is off while the temperature is not high, then it is turned on. It stays on while the temperature is not low.
- Reaction, $\tilde{u}_r$: This variable indicates that the reaction has started; although it is a property of the system it is treated as an (uncontrollable) control variable.

A summary of the controller output symbols is given in Table 11.4. Observe how

Symbol	Domain	Interpretation	Associated state
$\tilde{u}_b$	$\mathbb{B}$	turn on blender	$\tilde{q}_b$
$\tilde{u}_i$	$\mathbb{B}$	open inflow valve	$\tilde{q}_i$
$\tilde{u}_d$	$\mathbb{B}$	open draining valve	$\tilde{q}_d$
$\tilde{u}_h$	$\mathbb{B}$	turn on heater	$\tilde{q}_h$
$\tilde{u}_c$	$\mathbb{B}$	turn on cooler	$\tilde{q}_c$
$\tilde{u}_r$	$\mathbb{B}$	(start reaction)	$\tilde{q}_r$

Table 11.4: Outputs, and associated states, of the reactor controller.

we treat the "reaction control symbol" as any other control, the only difference is of course that its value is entirely determined by the system.

We now introduce a discrete state for each output of the controller. These controller states are shown in the last column of Table 11.4, they allow us to specify the controller output solely in terms of internal states and hence serve as our mechanism for keeping track of the continuous dynamics associated with each region. In addition, we need to define the events we wish to use for updating the controller states, these events specify the thresholds between "very low"/"not very low" level, etc.., and are summarized in Table 11.5. We then define a transition

Symbol	Domain	Threshold
$\tilde{e}_1$	$\mathbb{B}$	tank empty
$\tilde{e}_2$	$\mathbb{B}$	level very low
$\tilde{e}_3$	$\mathbb{B}$	level low
$\tilde{e}_4$	$\mathbb{B}$	level high
$\tilde{e}_5$	$\mathbb{B}$	tank full
$\tilde{e}_6$	$\mathbb{B}$	temperature below min
$\tilde{e}_7$	$\mathbb{B}$	no reaction
$\tilde{e}_8$	$\mathbb{B}$	temperature low
$\tilde{e}_9$	$\mathbb{B}$	temperature high
$\tilde{e}_{10}$	$\mathbb{B}$	temperature above max

Table 11.5: Inputs to the reactor controller.

relation for each state, updating it in response to the events according to the control rules stated above. For instance, the blender should be turned on when the level is not very low. Assuming that, initially, the level is very low, we assign the controller state $\tilde{q}_b$ the initial value `false` and then update the state according to

$$\tilde{q}_b^+ = \tilde{q}_b \wedge \neg \tilde{e}_2 \vee \neg \tilde{q}_b \wedge \tilde{e}_2 \tag{11.6}$$

expressing that the state should remain unchanged unless the event $\tilde{e}_2$ becomes `true` (level changes from "very low" to "not very low" or vice versa). The blender control signal then equals the blender state, i.e., $\tilde{u}_b = \tilde{q}_b$. This corresponds to the FSM in Figure 11.2. The outputs $\tilde{u}_d$, $\tilde{u}_h$ and $\tilde{u}_r$ can be obtained in a similar fashion. The outputs $\tilde{u}_i$ and $\tilde{u}_c$ differ, however, in that they involve hysteresis. For instance, we can obtain an implementation of the control strategy for $\tilde{u}_i$ using the FSM in Figure 11.3 which corresponds to the transition relation

$$\tilde{q}_i^+ = \tilde{q}_i \wedge \neg(\tilde{e}_3 \vee \tilde{e}_4) \vee \tilde{e}_3 \tag{11.7}$$

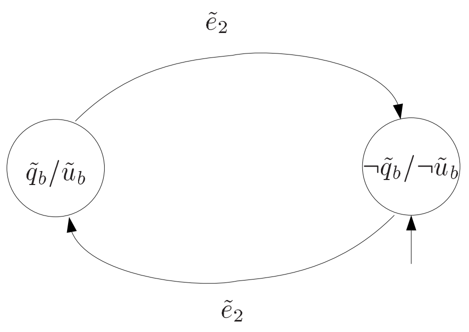


Figure 11.2: The finite state machine for the blender controller.

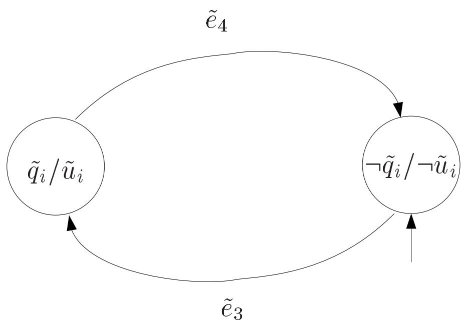


Figure 11.3: The finite state machine for the inflow valve controller.

Summing up, the events generated force the controller to update its state according to the controller transition relation

$$k(\tilde{q},\tilde{e}) = \begin{pmatrix} \tilde{q}_b \wedge \neg\tilde{e}_2 \vee \neg\tilde{q}_b \wedge \tilde{e}_2 \\ \tilde{q}_i \wedge \neg(\tilde{e}_3 \vee \tilde{e}_4) \vee \tilde{e}_3 \\ \tilde{q}_d \wedge \neg\tilde{e}_7 \vee \neg\tilde{q}_d \wedge \tilde{e}_7 \\ \tilde{q}_h \wedge \neg\tilde{e}_7 \vee \neg\tilde{q}_h \wedge \tilde{e}_7 \\ \tilde{q}_c \wedge \neg(\tilde{e}_9 \vee \tilde{e}_8) \vee \tilde{e}_9 \\ \tilde{q}_r \wedge \neg\tilde{e}_7 \vee \neg\tilde{q}_r \wedge \tilde{e}_7 \end{pmatrix} \tag{11.8}$$

while the output is defined using $l(\tilde{q}) = (\tilde{q}_b, \tilde{q}_i, \tilde{q}_d, \neg\tilde{q}_h, \tilde{q}_c, \tilde{q}_r)^T$

Although we know how we want the controller to behave, we still have to define what we mean by "high level", "low temperature" etc.. We thus need to design the interface between the logic controller and the continuous states, i.e., the generator.

Actuator

The actuator is implemented in a straightforward manner according to (5.10) resulting in an actuator function translating symbols $\tilde{u}_*$ to control signals u_*.

$$\begin{aligned}\alpha(\tilde{u}) = \{u \in \mathbb{R}^m \mid\ & \big[\tilde{u}_b \to [u_b = 1]\big] \wedge \big[\neg\tilde{u}_b \to [u_b = 0]\big] \wedge \\ & \big[\tilde{u}_i \to [u_i = 1]\big] \wedge \big[\neg\tilde{u}_i \to [u_d = 0]\big] \wedge \\ & \big[\tilde{u}_h \to [u_h = 1]\big] \wedge \big[\neg\tilde{u}_h \to [u_h = 0]\big] \wedge \\ & \big[\tilde{u}_c \to [u_c = 1]\big] \wedge \big[\neg\tilde{u}_c \to [u_c = 0]\big] \wedge \\ & \big[\tilde{u}_r \to [u_r = 1]\big] \wedge \big[\neg\tilde{u}_r \to [u_r = 0]\big]\}\end{aligned} \qquad (11.9)$$

We thus obtain a definition of the matrices $A(\tilde{u}), b(\tilde{u})$ for each assignment of values to the discrete control vector.

Generator

In this application, it becomes very clear that defining the generator can be a demanding design task. This can be contrasted with the discrete control law design which is obtained quite naturally due to our intuition suggesting that, for instance, we should turn on the cooler when the temperature gets too high, etc..

We do our best, though, of performing this task, the result is the hyperplanes shown in Table 11.6 along with the associated events.

Hyperplane	Definition	Associated event
$\mathcal{H}_1$	$x_1 = 0$	$\tilde{e}_1$
$\mathcal{H}_2$	$x_1 = 3$	$\tilde{e}_2$
$\mathcal{H}_3$	$25x_1 + x_2 = 250$	$\tilde{e}_3$
$\mathcal{H}_4$	$25x_1 + x_2 = 300$	$\tilde{e}_4$
$\mathcal{H}_5$	$x_1 = 13$	$\tilde{e}_5$
$\mathcal{H}_6$	$x_2 = 0$	$\tilde{e}_6$
$\mathcal{H}_7$	$x_2 = 50$	$\tilde{e}_7$
$\mathcal{H}_8$	$x_2 = 110$	$\tilde{e}_8$
$\mathcal{H}_9$	$x_2 = 130$	$\tilde{e}_9$
$\mathcal{H}_{10}$	$x_2 = 150$	$\tilde{e}_{10}$

Table 11.6: Hyperplanes defining the generator for the reactor.

Note that the definition of "low" and "high" fluid level depends on the temperature, the idea is that high temperature may be less desirable at higher fluid levels.

Remark 11.2
The inherent difficulties of deciding on the appropriate definition of "high temperature" etc.. suggest that a step towards controller synthesis would be to parameterize

some of the hyperplanes, keeping the parameterization throughout the verification procedure, and obtain parameter sets which conserve the verified properties.

An example trajectory obtained by simulating the hybrid model defined above is shown in Figure 11.4 along with the partition of the state space according to the generator. This trajectory might confirm with our ideas of normal operation, i.e.,

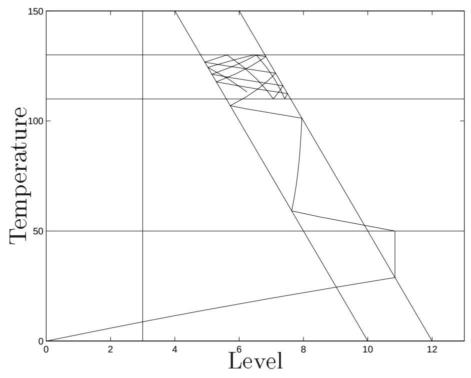


Figure 11.4: An example trajectory.

we would like the trajectory to gradually approach the operating region without exceeding allowed values on the way. In order to check if this is always the case, we apply model checking.

11.1.3 Verification Results

We intend to use this example to illustrate how model checking can be applied to the abstract system description using explicit state enumeration. Let us first take a look at what the specifications are in terms of bad states and goal states.

Specifications

The desired properties of the reactor stated in Section 11.1.1 translate into the following sets of bad and goal states.

- The temperature is never below minimum and never above maximum. This results in a definition of bad states

$$\mathcal{B}_T = \{\tilde{x} \in \mathcal{X} \mid [\tilde{x}_{10} = 1] \vee [\tilde{x}_6 = -1]\} \tag{11.10}$$

- The tank is never full and never empty. We thus get another set of bad states

$$\mathcal{B}_h = \{\tilde{x} \in \mathcal{X} \mid [\tilde{x}_1 = -1] \vee [\tilde{x}_5 = 1]\} \tag{11.11}$$

Together these sets of bad states form a specification of regions which should never be reached from any other region in the continuous state space. That is, we define a set of initial states as all other discrete states.

$$\mathcal{I}_1 = \{\tilde{x} \in \mathcal{X} \mid \neg(B_T(\tilde{x}) \vee B_h(\tilde{x}))\} \tag{11.12}$$

We then need to ensure that

$$(\mathcal{B}_T \cup \mathcal{B}_h) \cap \mathcal{R}^+ = \emptyset \tag{11.13}$$

where the reachable set $\mathcal{R}^+$ is obtained as the least fixed point starting from $\mathcal{I}_1$ and iterating the acceptor relation.

The third safety specification can be evaluated in a similar manner.

- If the temperature is not high and not low while the level is not high and not low, then the temperature and level will always lie in this region.

That is, we have a set of bad states

$$\mathcal{B} = \{\tilde{x} \in \mathcal{X} \mid \neg([\tilde{x}_8 = 1] \wedge [\tilde{x}_9 = -1] \wedge [\tilde{x}_3 = 1] \wedge [\tilde{x}_4 = -1])\} \tag{11.14}$$

and we need to ensure that these states are never reached from the initial states

$$\mathcal{I}_2 = \{\tilde{x} \in \mathcal{X} \mid [\tilde{x}_8 = 1] \wedge [\tilde{x}_9 = -1] \wedge [\tilde{x}_3 = 1] \wedge [\tilde{x}_4 = -1]\} \tag{11.15}$$

where we again use the acceptor when obtaining the set of reachable states.

Finally, we have a planning property to verify.

- The operating region should always be reachable from the initial states.

Now, we have not stated which states serve as initial states in this aspect. We can, for instance, select an initial state corresponding to the simulation in Figure 11.4.

$$\mathcal{I} = \{\tilde{x} \in \mathcal{X} \mid [\tilde{x}_1 = 1] \wedge [\tilde{x}_2 = -1] \wedge [\tilde{x}_6 = 1] \wedge [\tilde{x}_7 = -1]\} \tag{11.16}$$

We then obtain the set of states reachable states from this state as the least fixed point starting from $\mathcal{I}$ and using transitions which are accepted, not excepted and not reflected. We then check if the goal state

$$\mathcal{G} = \{\tilde{x} \in \mathcal{X} \mid [\tilde{x}_8 = 1] \wedge [\tilde{x}_9 = -1] \wedge [\tilde{x}_3 = 1] \wedge [\tilde{x}_4 = -1]\} \tag{11.17}$$

is a subset of the reachable states.

Results

Let us start by revealing that application of the proposed verification methods proves all the properties stated in the previous section to be `true`. We will now take our best shot at presenting these results via graphical representations of the reachable state spaces using different types of discrete approximations.

We start by examining transitions which are accepted by the switched system. These are shown in Figure 11.5 where each large circle represents an open region in the continuous state space and each small circle represents an open line segment separating the regions. We have thus left out discrete states corresponding to corner points since these are not of interest during our analysis. Furthermore, we have duplicates of the states representing regions where the controller implements hysteresis, since we have different dynamics for each controller state. For illustration purposes, we have emphasized a path corresponding to the simulation in Figure 11.4, this should aid in understanding the coupling between the dynamics of the switched system and our discrete approximation.

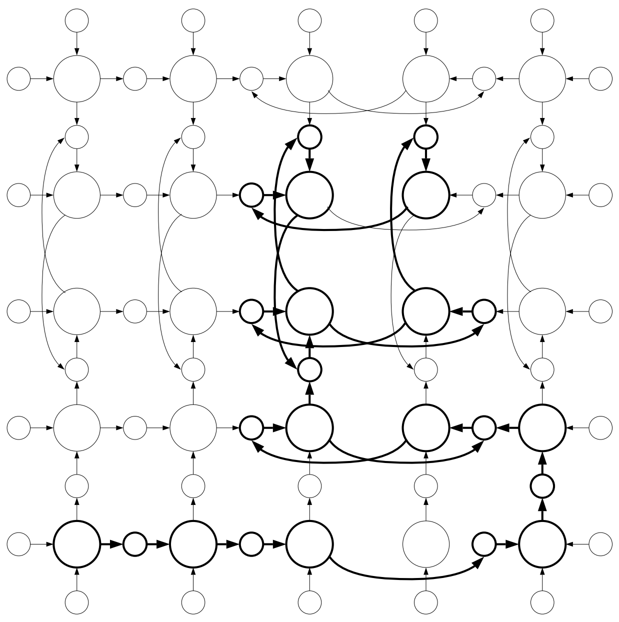


Figure 11.5: A transition diagram of the acceptor FSM for the reactor. Thin lines show possible states and transitions while thick lines show the path corresponding to the simulation in Figure 11.4.

We see in Figure 11.5 that the states representing intermediate edges provide no additional information. We may thus exclude these states from our graphical representation, this results in the somewhat clearer transition diagram shown in Figure 11.6 where we have marked the states corresponding to the operating region. Note that the operating region is shown as four different discrete states due to the dynamics of the controller which defines four different affine systems for the same region.

Let us now examine the transition relations shown in Figure 11.6. Indeed, we find that there are no transitions to the states corresponding to temperatures

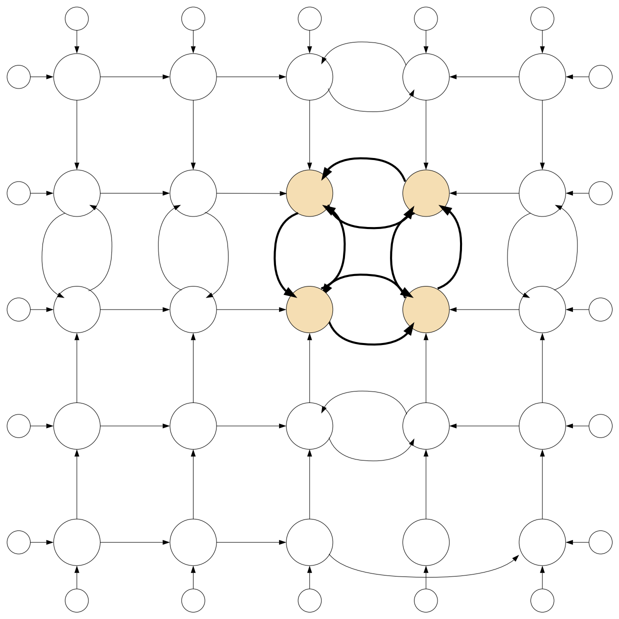


Figure 11.6: A reduced version of the transition diagram of the acceptor FSM for the reactor. Marked circles show the discrete states corresponding to the operating region.

and levels at max or min values (the small circles). We thus know that those transitions do not exist in the switched system. Furthermore, we see that there are no transitions out of the states defining the operating region. Hence we can also guarantee the third safety specification.

Turning to the planning property, we start by obtaining a transition diagram showing all transitions which are accepted and not excepted. This is shown in Figure 11.7. We see that the only transitions excepted are the ones from the operating region, all other regions have a bounding facet which attracts the state trajectory for all dynamics allowed by the controller. Now, this discrete model does not allow us to guarantee that the operating region will be reached from the initial state. This is due to the existence of loops involving the hysteresis regions and the bounding edges. But now we benefit from the use of reflected transitions, these are emphasized in Figure 11.7 and excluding them from our model, as well as neglecting states corresponding to transient edges, results in Figure 11.8. Clearly, the exclusion of reflected transitions enables us to guarantee that the operating region will be reached in finite time, not only from the specified initial state, but from any state which does not classify as a bad state. This is also implied when we consider an extensive simulation of the switched system, starting from several points within every region. These simulation results are shown in Figure 11.9.

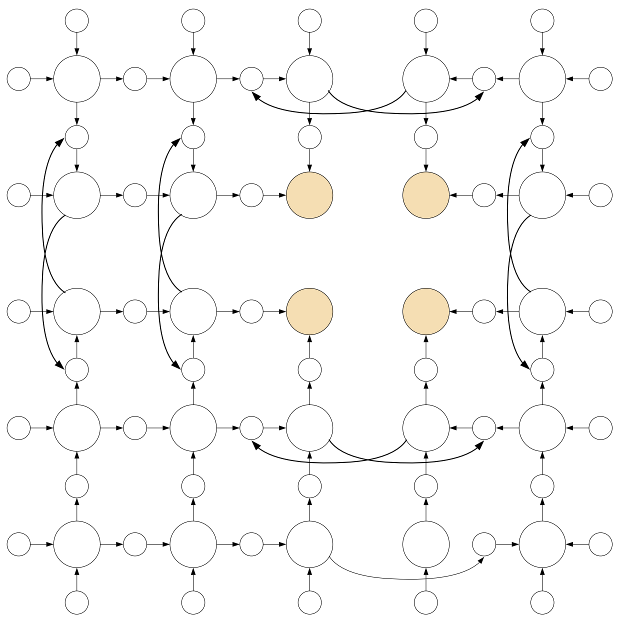


Figure 11.7: The transition diagram for accepted and not excepted transitions for the reactor. Marked circles show the discrete states corresponding to the operating region and emphasized arcs show reflected transitions.

11.2 Landing Gear

In this section we study the landing gear system of the Swedish JAS 39 Gripen defense and fighter aircraft depicted in Figure 11.10.

This system is treated in detail in [45] where models of the system and controller are obtained using switched bond graphs [59] at the physical level and hybrid transition systems [44] at the mathematical level. The models obtained are then used to illustrate verification techniques based both on extended duration calculus [10] and on the formalism of hybrid automata [29], [1], using the HyTech model checker [30]. A description of the system and a detailed description of the controller can also be found in [22] where symbolic model checking is used to verify the function specifications of the landing gear controller.

We also intend to use this application to illustrate the methods derived in the previous chapters. We start by discussing the system at hand and the specifications it should satisfy. We then obtain a system model and illustrate simulation results which provide some intuition about the system behavior. We then obtain our discrete devices as defined in Chapter 9 and perform the verification.

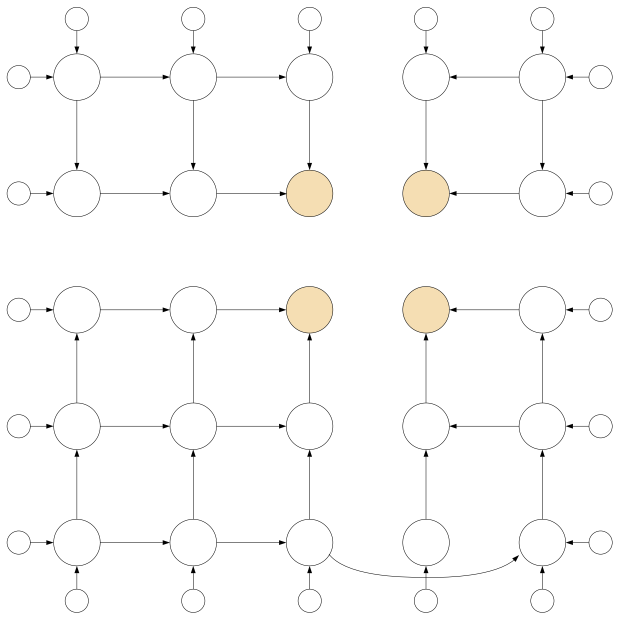


Figure 11.8: The transition diagram for accepted, not excepted and not reflected transitions for the reactor.

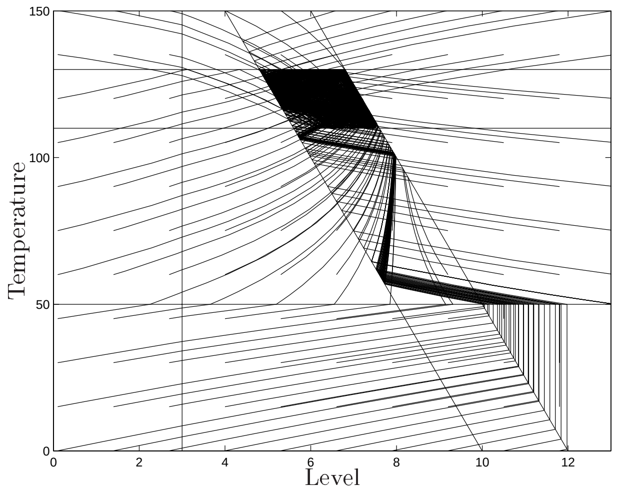


Figure 11.9: Results from extensive simulation of the switched model.

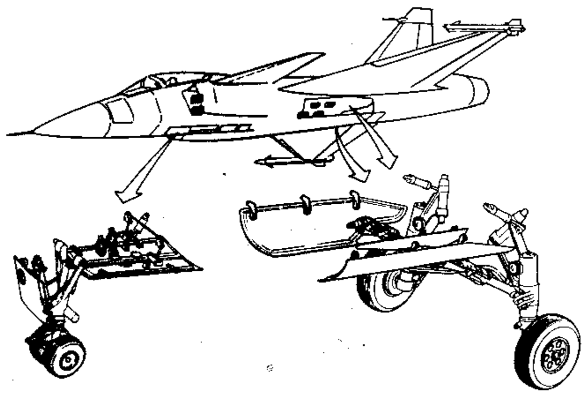


Figure 11.10: The landing gear of JAS 39 Gripen.

11.2.1 System Description

The landing gear system consists of three landing gears with corresponding doors and a landing gear controller implemented in Pascal. The gears and doors are operated using hydraulic actuators, one for each gear and door. However, neither the gears nor the doors can be manipulated individually but they are operated in parallel. The hydraulic pressure in the actuators is maintained using a hydraulic power supply system.

The landing gear is operated electrically via a lever in the cockpit issuing extension or retraction commands to the landing gear controller. In addition, emergency extension can be commanded to an emergency controller implemented in hardware logic. In that case, the gear is extracted using the air drag and not the hydraulic power.

The software controller is a complex discrete event dynamic system deserving detailed analysis on its own, see [22]. We thus choose to illustrate our methods for the case when the landing gear is controlled using the simpler hardware controller, assuming that the hydraulic system is used for operating the gear also in this case.

The specifications we wish to verify are twofold:

- The doors and gear should not collide during extension or retraction of the landing gear.

- When an extension or a retraction command is issued, the landing gear should extend or retract, respectively, and the doors close, in finite time.

The first specification clearly falls into the safety category, while the second classifies as a goal state problem. This suggests that we may be able to use our discrete devices to verify the specifications. First, though, we need a mathematical model of the system.

11.2.2 System Model

This section deals with modeling of the landing gear system. We will, for clarity reasons, consider a model for a single door and gear; since all doors and gears are operated in parallel, we do not gain anything from distinguishing each subsystem.

Plant

Starting with the door position, activating the signal u_{od}, causes the hydraulic valve to open for flow into the extension side of the hydraulic cylinder manipulating the door. Similarly, setting $u_{cd} = 1$, causes the valve to direct the flow into the retraction side of the cylinder. The corresponding change in door position will then either be positive or negative, respectively, and the rate of change depends on the hydraulic supply pressure. The above can be modeled using the following differential equation for door position, x_d.

$$\dot{x}_d = \alpha_d x_p (u_{od} - u_{cd}) \tag{11.18}$$

where x_p is the supply pressure and α_d is a positive constant.

Analogously, we get a model of the change in gear position, x_g,

$$\dot{x}_g = \alpha_g x_p (u_{eg} - u_{rg}) \tag{11.19}$$

since $u_{eg} = 1$ opens the valve for flow into the extension side of the gear cylinder and $u_{rg} = 1$ results in flow into the retraction side. Again, $\alpha_g > 0$ is a constant.

It only remains to model the change in hydraulic pressure. It can be captured by a first order affine differential equation where the time constant (and hence the stationary point) depends on the number of open valves. That is,

$$\begin{aligned} \dot{x}_p = -(\gamma + \gamma_d(u_{od} + u_{cd} - 2u_{od}u_{cd}) +& \\ \gamma_g(u_{eg} + u_{rg} - 2u_{eg}u_{rg}))x_p + \beta& \end{aligned} \tag{11.20}$$

where $\gamma, \gamma_d, \gamma_g$ and β are positive constants.

Remark 11.3
Note that we might here take advantage of the fact that the commands activating the signals u_{od} and u_{cd} signals are mutually exclusive, as are the commands activating u_{eg} and u_{rg}. This would lead to a slightly simpler plant model (replace $(u_{od} + u_{cd} - 2u_{od}u_{cd})$ by $(u_{od} + u_{cd})$) but violates the separation of plant and controller into independent entities.

A summary of signals is provided in Tables 11.7 and 11.8.

In the sequel, we will assume that the parameters have the values $\alpha_d = \alpha_g = \gamma = \gamma_d = \gamma_g = 1$ and $\beta = 4$. Introducing the state vector $x = (x_d, x_g, x_p)^T$ and control vector $u = (u_{od}, u_{cd}, u_{eg}, u_{rg})^T$, we get the affine state space model

Signal	Domain	Interpretation
u_{od}	$\{0,1\}$	open door valve signal
u_{cd}	$\{0,1\}$	close door valve signal
u_{eg}	$\{0,1\}$	extend gear valve signal
u_{rg}	$\{0,1\}$	retract gear valve signal

Table 11.7: Inputs to the landing gear plant model.

Signal	Domain	Interpretation
x_g	$\mathbb{R}$	door position
x_d	$\mathbb{R}$	gear position
x_p	$\mathbb{R}$	hydraulic pressure

Table 11.8: States (and outputs) of the landing gear plant model.

$\dot{x} = A(u)x + b(u)$ with

$$A(u) = \begin{pmatrix} 0 & 0 & u_{od} - u_{cd} \\ 0 & 0 & u_{eg} - u_{rg} \\ 0 & 0 & -1 - (u_{od} + u_{cd} - 2u_{od}u_{cd}) - (u_{eg} + u_{rg} - 2u_{eg}u_{rg}) \end{pmatrix} \tag{11.21}$$

$$b(u) = \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix}$$

Note that during extension or retraction of door or gear, we obtain a model where the corresponding position increases or decreases without bounds. However, due to the discrete nature arising from physical limits on possible positions, this will not be the case in the actual system. Furthermore, it seems natural that a controller is implemented which takes actions when we approach the hard physical limits, thus defining new "soft" limits. Let us next take a closer look at the actual controller implementation.

Controller

As explained in Section 11.2.1, we will examine the system when controlled by the hardware controller intended for emergency operation.

The controller has access to discrete symbols indicating occurrence of events in the system, it then produces output symbols which are translated into corresponding actuator signals. There is no explicit need for internal states since the controller is static, i.e., it does not utilize any memory. However, since we prefer to model the events as the occurrence of a hyperplane crossing, irrespective of direction, we define a state for each event used by the controller and then define the output in terms of the current controller state only. A summary of the controller symbols can be found in Tables 11.9 and 11.10.

Symbol	Domain	Threshold	Associated state
$\tilde{e}_{dc}$	$\mathbb{B}$	door closed	$\tilde{q}_{dc}$
$\tilde{e}_{do}$	$\mathbb{B}$	door open	$\tilde{q}_{do}$
$\tilde{e}_{gi}$	$\mathbb{B}$	gear in	$\tilde{q}_{gi}$
$\tilde{e}_{go}$	$\mathbb{B}$	gear out	$\tilde{q}_{go}$
$\tilde{e}_{pl}$	$\mathbb{B}$	pressure low	-
$\tilde{e}_{ph}$	$\mathbb{B}$	pressure high	-
$\tilde{e}_{cmd}$	$\mathbb{B}$	pilot command	$\tilde{q}_{cmd}$

Table 11.9: Inputs, and associated states, to the landing gear controller.

Symbol	Domain	Interpretation
$\tilde{u}_{od}$	$\mathbb{B}$	open door
$\tilde{u}_{cd}$	$\mathbb{B}$	close door
$\tilde{u}_{eg}$	$\mathbb{B}$	extend gear
$\tilde{u}_{rg}$	$\mathbb{B}$	retract gear

Table 11.10: Outputs of the landing gear controller.

Remark 11.4
In Section 11.1, we defined a state for each output instead of each input event as we do here. The latter approach may seem more natural, since we introduce the memory in order to keep track of on which side of a hyperplane the trajectory lies. However, in the reactor case, it sufficed to introduce a state for each output and we were thus able to implement a controller with slightly fewer internal states.

Note that the description of the event threshold can be seen as an interpretation of the associated controller state symbol. Generally, the interpretation of state and controller symbols refers to the case when the corresponding symbol is `true`, a `false` symbol thus refers to the negation of the interpretating statement. For instance, $\tilde{q}_{dc} =$ `false` would mean that the door is not closed and $\tilde{u}_{eg}$ means that we do not want to extend the gear. An exception from this rule is the symbol $\tilde{q}_{cmd}$ which represents the pilot command. A `true` value represents the pilots intention to start the procedure of extending the landing gear (open door, extend gear, close door) while the value `false` commands starting the retraction procedure (open door, retract gear, close door).

The controller behavior can now be stated in terms of a logical function mapping controller states to outputs

$$\begin{aligned} \tilde{u}_{od} &= (\tilde{q}_{cmd} \wedge \neg\tilde{q}_{go} \ \vee \ \neg\tilde{q}_{cmd} \wedge \neg\tilde{q}_{gi}) \wedge \neg\tilde{q}_{do} \wedge \neg\tilde{u}_{cd} \\ \tilde{u}_{cd} &= (\tilde{q}_{cmd} \wedge \tilde{q}_{go} \ \vee \ \neg\tilde{q}_{cmd} \wedge \tilde{q}_{gi}) \wedge \neg\tilde{q}_{dc} \wedge \neg\tilde{u}_{od} \\ \tilde{u}_{eg} &= \tilde{q}_{cmd} \wedge \tilde{q}_{do} \wedge \neg\tilde{q}_{go} \wedge \neg\tilde{u}_{rg} \\ \tilde{u}_{rg} &= \neg\tilde{q}_{cmd} \wedge \tilde{q}_{do} \wedge \neg\tilde{q}_{gi} \wedge \neg\tilde{u}_{eg} \end{aligned} \tag{11.22}$$

where the controller states are updated using a transition relation obtained as

$$\tilde{q}_*^+ = \tilde{q}_* \wedge \neg\tilde{e}_* \ \vee \ \neg\tilde{q}_* \wedge \tilde{e}_* \tag{11.23}$$

where the subscript $*$ can be replaced with any of the corresponding states and events.

The output function is implemented using standard logical components, a variant implemented in the modeling and simulation environment Simulink is shown in Figure 11.11. Note that, for simulation purposes, we do not need to introduce

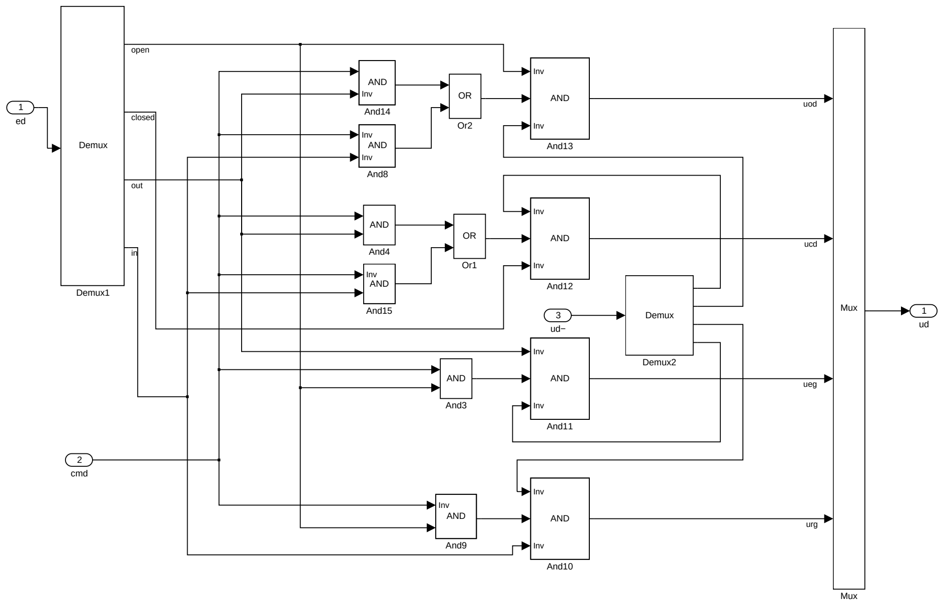


Figure 11.11: The landing gear output function.

explicitly the controller states as we did above. This is due to the signal interpretation of variables in the simulation environment. Compare the discussion of signals vs. events in [22].

Actuator

The actuator is implemented in a straightforward manner according to (5.10) resulting in an actuator function translating symbols $\tilde{u}_*$ to control signals u_*.

$$\begin{aligned}\alpha(\tilde{u}) = \{u \in \mathbb{R}^m \mid\ & \big[\tilde{u}_{od} \to [u_{od} = 1]\big] \wedge \big[\neg\tilde{u}_{od} \to [u_{od} = 0]\big] \wedge \\ & \big[\tilde{u}_{cd} \to [u_{cd} = 1]\big] \wedge \big[\neg\tilde{u}_{cd} \to [u_{cd} = 0]\big] \wedge \\ & \big[\tilde{u}_{eg} \to [u_{eg} = 1]\big] \wedge \big[\neg\tilde{u}_{eg} \to [u_{eg} = 0]\big] \wedge \\ & \big[\tilde{u}_{rg} \to [u_{rg} = 1]\big] \wedge \big[\neg\tilde{u}_{rg} \to [u_{rg} = 0]\big]\}\end{aligned} \tag{11.24}$$

We thus obtain a definition of the matrices $A(\tilde{u}), b(\tilde{u})$ for each assignment of values to the discrete control vector.

Generator

Specifying the generator means defining the hyperplanes which are responsible for the event generation and, later on, the discretization of our model.

Four hyperplanes are obtained quite naturally from the (soft) limits on possible positions of door and gear. We supplement these by defining bounds on the pressure; this is convenient when considering the discretization since we will then require that certain regions defined by the hyperplanes are bounded.

This results in the hyperplanes shown in Table 11.11 along with the associated events.

Hyperplane	Definition	Associated event
$\mathcal{H}_{dc}$	$x_d = 0$	$\tilde{e}_{dc}$
$\mathcal{H}_{do}$	$x_d = 1$	$\tilde{e}_{do}$
$\mathcal{H}_{gi}$	$x_g = 0$	$\tilde{e}_{gi}$
$\mathcal{H}_{go}$	$x_g = 1$	$\tilde{e}_{go}$
$\mathcal{H}_{pl}$	$x_p = 0$	$\tilde{e}_{pl}$
$\mathcal{H}_{ph}$	$x_p = 10$	$\tilde{e}_{ph}$

Table 11.11: Hyperplanes defining the generator.

Note that there is no hyperplane defined for the event $\tilde{e}_{cmd}$ since this is an external control symbol associated with an external event.

We have now obtained a complete model of our system, this model is in accordance with the framework described in Section 5.2 and we can thus proceed directly to discretization of the model. Of course, the model obtained can also be used for simulation, thus providing some intuition regarding the behavior of the system. A SIMULINK implementation is shown in Figure 11.12 and signal diagrams for a simulation where, initially, the door is closed, gear is in, the pressure is not in steady state, and the pilot command is issued at $t = 1s$, are shown in Figures 11.13 and 11.14.

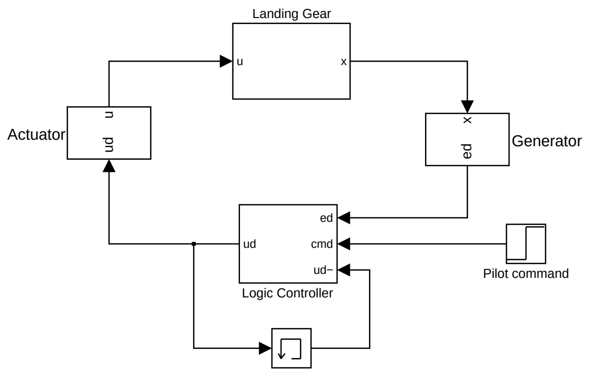


Figure 11.12: The landing gear simulation model.

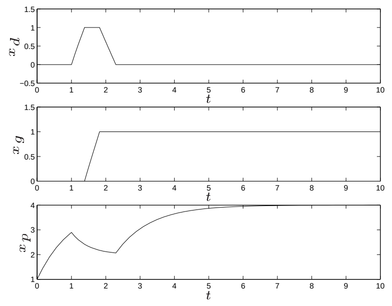


Figure 11.13: State variables.

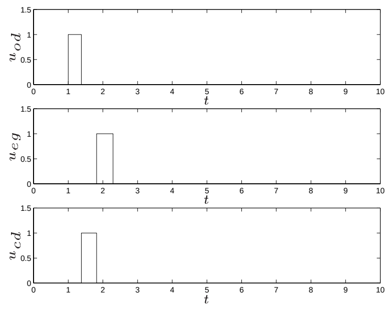


Figure 11.14: Control signals.

In Figure 11.15 we provide additional example trajectories, again starting from points where the door is closed and gear in, but now the pilot command is issued at different time points, resulting in several different trajectories. However, we plot the trajectories in the continuous state space instead of viewing them as explicit functions of time. We clearly see that all trajectories display similar behavior. When the command is issued, the door opens while the gear position is unchanged and the pressure moves towards its steady state value (here $\bar{x}_p = 2$). When the door is open, the gear is extended while the door is stationary, this proceeds until the gear reaches its end position. Then the doors start closing, the gear being stationary, and when the doors are closed the pressure reaches steady state (now $\bar{x}_p = 4$).

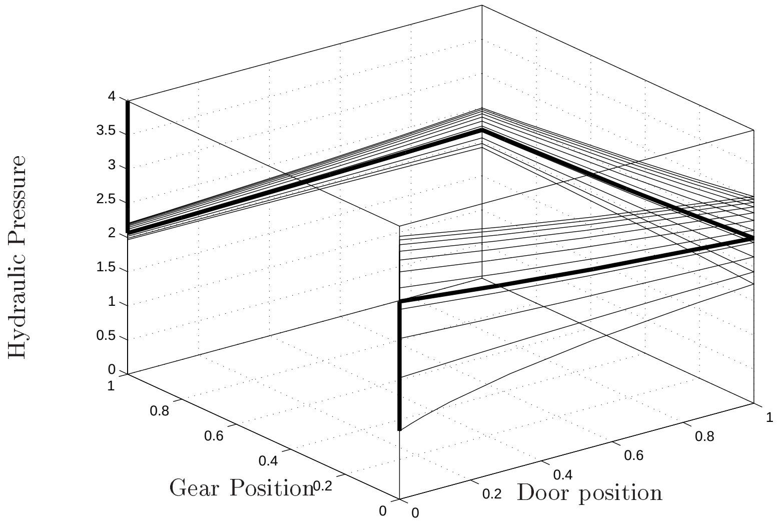


Figure 11.15: Simulated trajectories of the landing gear model.

This analysis via simulation provides us with some confidence about the model behavior. It is clear, though, that we have only covered a small portion of possible cases; a multitude of different initial conditions, initial controller states etc.. need to be considered before we can be confident in that the model satisfies the desired properties. Hence the need for formal methods.

11.2.3 Verification Results

We now illustrate the results from a verification procedure involving temporal logic specifications of desired properties, and the use of the discrete devices constructed in Chapter 9.

Specifications

We have already stated the properties we are interested in verifying, namely that the doors and gear never collide, and that an extension command always results in a completed gear extension (the retraction case is analogous). More formally, utilizing temporal logic, we can write these specifications as:

- $S_B = \mathsf{EF}([\tilde{x}_{do} = -1] \wedge [\tilde{x}_{ge} = -1] \wedge [\tilde{x}_{gr} = 1])$
- $S_G = \mathsf{AG}(\tilde{e}_{cmd} \to \mathsf{AF}([\tilde{x}_{ge} = 0] \wedge [\tilde{x}_{dc} = 0]))$

The former condition states that there exist a path where the trajectory enters a critical region at some future time.

The latter condition states that all paths should be such that, in response to an extension command, the state trajectory always reaches the line defined by the intersection of $\mathcal{H}_{ge}$ and $\mathcal{H}_{dc}$ at some future time.

Let us now examine whether these conditions can be verified via the discrete approximations.

Results

We now consider transitions in a discrete state space obtained by the discretization procedure of Section 9.2. We define the set of initial states as the states corresponding to closed doors, retracted gear and hydraulic pressure in a normal operation interval. That is, the initial states are defined using

$$I(\tilde{x}) = [\tilde{x}_{dc} = 0] \wedge [\tilde{x}_{gr} = 0] \wedge [\tilde{x}_{pl} = 1] \wedge [\tilde{x}_{ph} = -1] \tag{11.25}$$

Starting with the safety specification, we have formulated it in terms of a temporal formula stating the existence of a computation path which, eventually, leads to a set of states specified by a Boolean formula. The evaluation of this temporal formula is a fixed point computation starting from the innermost Boolean formula, i.e.,

$$p = [\tilde{x}_{do} = -1] \wedge [\tilde{x}_{ge} = -1] \wedge [\tilde{x}_{gr} = 1] \tag{11.26}$$

All other states satisfying the temporal formula are then obtained by computing the least fixed point

$$\mathsf{EF}(p) = \mu_x(p \vee \mathsf{EX}(x)) = \mu_x(p \vee \gamma_1^-(M, x)) \tag{11.27}$$

where the transition relation M belongs to the acceptor since we are verifying a safety specification. Now, it turns out that the backward image of the states defined by p is empty, i.e., there do not exist any states which lead to the bad states in one step. Hence, the fixed point computation terminates after one step, yielding $\mathsf{EF}(p) = p$. The rest of the verification then consist of checking whether the this set intersects the set of initial states, this is of course not the case, i.e.,

$$\begin{aligned} \mathsf{EF}(p) \wedge I(\tilde{x}) = & [\tilde{x}_{do} = -1] \wedge [\tilde{x}_{ge} = -1] \wedge [\tilde{x}_{gr} = 1] \wedge \\ & [\tilde{x}_{dc} = 0] \wedge [\tilde{x}_{gr} = 0] \wedge [\tilde{x}_{pl} = 1] \wedge [\tilde{x}_{ph} = -1] \\ = & \texttt{false} \end{aligned} \tag{11.28}$$

It turns out that the trajectories are confined to the part of the state space corresponding to the sides of the cube shown in Figure 11.15. We visualize this state space by flattening out the sides, yielding a 2-dimensional representation of the regions involved. The result is visualized in Figure 11.16 and we thus get access to a transition diagram involving the states we find of interest, these correspond to the regions and line segments of Figure 11.16. The transition diagram is shown in Figure 11.17 where we have emphasized the state transitions corresponding to the simulation.

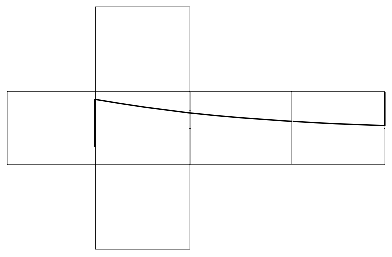


Figure 11.16: A 2-dimensional view of the sides of the cube in Figure 11.15 along with the trajectory from the simulation.

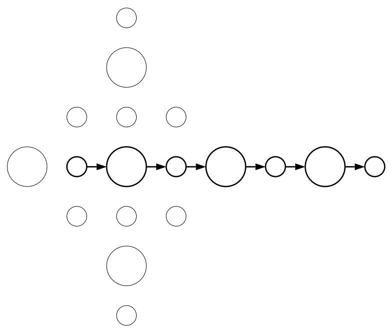


Figure 11.17: A transition diagram for the flattened regions of Figure 11.16. Transitions corresponding to the simulation of Figure 11.15 are emphasized.

We now need to show that we can always guarantee that trajectories starting in the initial state show the same behavior as we obtained in our simulation. That is, we need to evaluate the second specification formula and check if it is implied by the initial state formula.

We start by transforming the goal state formula into the equivalent temporal formula

$$S_G = \neg \mathsf{EF}(\tilde{e}_{cmd} \wedge \mathsf{EG}(\neg p)) \tag{11.29}$$

where p now specifies our goal state, i.e.,

$$p = [\tilde{x}_{ge} = 0] \wedge [\tilde{x}_{dc} = 0] \tag{11.30}$$

Now, starting with the innermost expression, we obtain

$$\mathsf{EG}(\neg p) = \nu_x(\neg p \wedge \mathsf{EX}(x)) = \nu_x(\neg p \wedge \gamma_1^-(M, x)) \tag{11.31}$$

where M should now express transitions which can be guaranteed to occur. Since our model does not contain any excepted transitions, this transition relation is the same as for the acceptor. The fixed point iteration now extracts the states leading to an infinite path not containing the goal states. Since there are no such states in our model, the expression yields the value `false`. We are therefore left with the formula

$$S_G = \neg\mathsf{EF}(\texttt{false}) \tag{11.32}$$

which has the value `true`. And since everything implies `true`, the formula defining the initial states does that in particular, and hence we can guarantee that the goal state will always be reached.

Note that the deductions described above are performed more or less manually, even though we have performed the discretization involved using quantifier elimination, we have not taken the full step towards utilizing BDD/IDD based model checking methods for verifying the above properties using these discretized models.

It should also be noted that, using the current tools available, the expressions involved in this example seem to be close to what can be handled using quantifier elimination. This is mainly due to the large number of discrete parameters which are kept throughout the calculations and the fact that the software is not designed with expressions involving such parameters in mind. Hence, examining alternative implementations taking these aspects into account may be a rewarding field for further study.

Bibliography

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] A.D. Baker, T.L. Johnson, D.I. Kerpelman, and H.A. Sutherland. GRAFCET and SFC as factory automation standards.

[4] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, USA, 1997.

[5] Richard E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, USA, 1983.

[6] M. S. Branicky. General hybrid dynamical systems: Modeling, analysis, and control. In R. Alur, T. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III – Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 186–200. Springer-Verlag, Berlin, 1996.

[7] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[8] J.R. Burch, E.M. Clarke, K.L. McMillan, D. Dill, and L.J.Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. of the 5th IEEE Symp. on Logic in Computer Science*, pages 428–439, Philadelphia, PA, USA, June 1990. IEEE Computer Society Press, Los Alamitos, CA, USA.

[9] F.E. Cellier. *Continuous System Modeling.* Springer-Verlag, Ann Arbor, MI, USA, first edition, 1991.

[10] Z. Chaochen, A. P. Ravn, and M. R. Hansen. An extended duration calculus for hybrid real-time systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *ProcẆorkshop on Theory of Hybrid Systems*, pages 36–59, Lyngby, Denmark, 1993. Springer-Verlag.

[11] A. Chutinan and B. H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. 1569:76–??, 1999. ISSN 0302-9743.

[12] E. Clarke and J. Wing. Formal methods: State of the art and future directions. Technical report, CMU Computer Science, 1996.

[13] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Dexter Kozen, editor, *Logic of Programs: Workshop*, volume 131 of *Lecture Notes in Computer Science.* Springer-Verlag, 1981.

[14] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[15] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the futurebus+ cache coherence protocol. In *Proceeding of the 11th International Symposium on Computer Hardware Description Languages and their Applications*, Ottawa, Ont., Canada, 1993.

[16] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symbolic Computation*, 12:299–328, 1991.

[17] Per-Erik Danielsson and Lennart Bengtsson. *Digital Teknik.* Studentlitteratur, Lund, Sweden, 1986.

[18] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[19] Andreas Dolzmann and Thomas Sturm. Simplification of quantifier-free formulae over ordered fields. *J. Symbolic Computation*, 24:209–231, 1997.

[20] K. Forsman. Hybrid control systems and comprehensive gröbner bases. In *Proceedings of the 33rd Conference on Decision and Control*, pages 4092–4097, Lake Buena Vista, FL., 1994.

[21] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics*. Addison Wesley, second edition, 1989.

[22] Johan Gunnarsson. *Symbolic Methods and Tools for Discrete Event Dynamic Systems*. PhD thesis, Dept. of Electrical Engineering, Linköping University, 1997.

[23] Aarti Gupta. Formal hardware verification methods: A survey. *The Journal of Formal Methods in Systems Design*, 1:151–238, 1992.

[24] N. Halbwachs, P. Raymond, and Y-E. Proy. Verification of linear hybrid systems by means of convex approximations. In *Proceedings of the First International Static Analysis Symposium*, pages 223–237, Namur, Belgium, 1994. Springer-Verlag.

[25] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[26] Simon Haykin. *Digital Communications*. John Wiley & Sons, Singapore, 1988.

[27] T. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In R. Alur, T. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III – Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 377–388. Springer-Verlag, Berlin, 1996.

[28] T. A. Henzinger and P.-H. Ho. A note on abstract interpretation strategies for hybrid automata. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 253–264. Springer-Verlag, Berlin, 1995.

[29] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 278–292, 1996.

[30] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: a model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1:110–122, 1997.

[31] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, USA, 1979.

[32] Mats Jirstrand. *Constructive Methods for Inequality Constraints in Control*. PhD thesis, Dept. of Electrical Engineering, Linköping University, 1998.

[33] Thomas Kailath. *Linear Systems*. Prentice-Hall, USA, 1980.

[34] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics, A Unified Approach*. Wiley Interscience, USA, 1990.

[35] Dean Kelley. *Automata and Formal Languages: An Introduction.* Prentice-Hall, USA, 1995.

[36] S. Kowalewski, S. Engell, J. Preußig, and O. Stursberg. Verification of logic controllers for continuous plants using timed condition/event-system models. *Automatica*, 35(3):505–518, mar 1999. URL `http://www-verimag.imag.fr/VHS/resources/KEPS99.pdf`. Special Issue on Hybrid Systems.

[37] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Transòn Computer-Aided Design*, 10(11):1356–1371, November 1991.

[38] Peter Lindskog. *Methods, algorithms and tools for system identification based on prior knowledge.* PhD thesis, Dept. of Electrical Engineering, Linköping University, 1996.

[39] L. Ljung. *System Identification: Theory for the User.* Prentice-Hall, sixth edition, 1987.

[40] L. Ljung. Non-linear black box modeling in system identification. In *Proceedings of the International Conference on Artificial Neural Networks*, Paris, France, October 1995.

[41] L. Ljung and T. Glad. *Modeling of Dynamic Systems.* Prentice-Hall, first edition, 1994.

[42] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[43] Kenneth L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[44] Simin Nadjm-Tehrani. Reactive systems in physical environments. In *Hybrid Problems, Hybrid Solutions - Procif the 10th Biennial Conference on AI and Cognitive Science*, pages 583–604, april 1995.

[45] Simin Nadjm-Tehrani and Jan-Erik Strömberg. Jas-95 lite: Modelling and formal analysis of dynamic properties. Technical report, Department of Computer and Information Science, Linköping University, 1996.

[46] H.M. Paynter. *Analysis and Design of Engineering Systems.* MIT Press, Cambridge, Mass., 1961.

[47] N. B. O. L. Pettit, T. Manavis, and P. E. Wellstaed. Using graph theory to visualise piecewise linear systems. In *Proceedings 3rd European Control Conference*, pages 1631–1636, Rome, Italy, September 1995.

[48] N. B. O. L. Pettit and P. E. Wellstaed. Piecewise-linear systems with logic control: A state space representation. In J.W. Nieuwenhuis, C. Praagman, and H.L. Trentelman, editors, *Proceedings 2nd European Control Conference*, pages 1581–1586, Groningen, Netherlands, 1993.

[49] N. B. O. L. Pettit and P. E. Wellstaed. A graphical analysis method for piecewise linear systems. In *Proceedings of the 33rd Conference on Decision and Control*, pages 1122–1127, Lake Buena Vista, FL, USA, December 1994.

[50] A. Pnueli. A temporal logic of concurrent programs. *Theor. Comp. Sci.*, 13: 45–60, 1981.

[51] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 358–369. Springer-Verlag, Berlin, 1995.

[52] P. Ramadge and W. Wonham. The control of discrete event systems. In *Proceedings of the IEEE*, pages 1073–1077, January 1989.

[53] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs on Theoretical Computer Science*. 1985.

[54] Wilsin J. Rugh. *Linear Systems Theory*. USA, 1996.

[55] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, AC-26(2):346–357, April 1981.

[56] R.S. Sreenivas and B.H. Krogh. On condition/event systems with discrete state realizations. *Discrete Event Dynamic Systems: Theory and Applications*, 1: 209–235, September 1991.

[57] J. Stiver, P. J. Antsaklis, and M. D. Lemmon. Interface and controller design for hybrid systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 462–492. Springer-Verlag, Berlin, 1995.

[58] J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon. An invariant based approach to the design of hybrid control systems containing clocks. In R. Alur, T. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III – Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 464–474. Springer-Verlag, Berlin, 1996.

[59] Jan-Erik Strömberg. *A mode switching modelling philosophy*. PhD thesis, Dept. of Electrical Engineering, Linköping University, 1994.

[60] Michael Tittus. *Control Synthesis for Batch Processes*. PhD thesis, School of Electrical and Computer Engineering, Chalmers University of Technology, 1995.

[61] F. Uhlig. A recurring theorem about pairs of quadratic forms and extensions: A survey. *Linear Algebra and Its Applications*, 25(219-237), 1979.

[62] M. Vidyasagar. *Nonlinear Systems Analysis.* Prentice-Hall, second edition, 1993.

[63] Volker Weispfenning. Simulation and optimization by quantifier elimination. *J. Symbolic Computation*, 24:189–208, 1997.

[64] G. M. Ziegler. *Lectures on Polytopes.* Graduate Texts in Mathematics. Springer-Verlag, New York, 1994.

Notation

Symbols

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{A}$	The sets of natural, integer, rational, and algebraic numbers.
$\mathbb{R}$, $\mathbb{C}$	The sets of real and complex numbers.
I	Identity matrix of appropriate dimension.
$\wedge$, $\vee$, $\neg$, $\rightarrow$	Boolean conjunction, disjunction, negation, and implication.
$\exists$, $\forall$	Existential and universal quantifier.
$\emptyset$	The empty set.
∂M	Boundary of set M.

Operators and Functions

$A \subseteq B$	A is a subset of B.
$A \subsetneq B$	A is a proper subset of B.
$A \cap B$	Intersection of sets A and B.
$A \cup B$	Union of sets A and B.
$A \setminus B$	The set $\{x \mid x \in A \text{ and } x \notin B\}$.
∂	Derivation operator.

$\dot{x}, \ddot{x}, x^{(i)}$	First, second, and i^{th} derivative of x w.r.t. t.
$P \succ 0$, $P \succeq 0$	Positive (semi-)definite matrix.
$P \prec 0$, $P \preceq 0$	Negative (semi-)definite matrix.
M^T	Transpose of matrix M.
M^{-1}	Inverse of matrix M.
$M^{\dagger}$	Generalize (or Moore-Penrose) inverse of matrix M.
$\det(M)$	Determinant of matrix M.
$\log\det(M)$	Logarithm of the determinant of matrix M.
$M_{\perp}$	Full rank matrix such that $M_{\perp} M = 0$.
$\mathcal{N}(M)$	Null space of matrix M.
$\mathcal{R}(M)$	Range space of matrix M.
$M_{\mathcal{N}}$	A matrix whose columns form a basis of $\mathcal{N}(M)$.
$\operatorname{argmin}_x f(x)$	Minimizing argument of $f(x)$.
$\operatorname{vol}(\mathcal{E})$	Volume of an ellipsoid $\mathcal{E}$.
$\mathbf{Co}\{v_1, \ldots, v_p\}$	Convex hull of the vectors $\{v_1, \ldots, v_p\}$.
$V_x(x)$	Row vector of partial derivatives of $V(x)$ w.r.t. $x_1, \ldots, x_n$.
$\lvert x \rvert$	Euclidean norm of a vector x.

Acronyms

LMI	Linear Matrix Inequality.
LP	Linear Programming.
SDP	Semidefinite Programming.
MAXDET	Determinant Maximization.
QE	Quantifier Elimination.

PhD Dissertations, Division of Automatic Control, Linköping University