Formal Verification of Continuous Systems

Sloth, Christoffer

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Christoffer Sloth

Formal Verification of Continuous Systems

Formal Verification of Continuous Systems
PhD dissertation

July 2012

Contents

Preface and Acknowledgements

This dissertation is submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy from Department of Computer Science at Aalborg University. The work was carried out in the period from August 2009 until July 2012 under careful supervision of Professor Rafael Wisniewski. The members of the assessment committee were Professor Antoine Girard, Université Joseph Fourier, France, Professor John Lygeros, Eidgenössische Technische Hochschule Zürich, Switzerland, and Professor Anders P. Ravn, Aalborg University, Denmark.

First and foremost, I would like to thank my supervisor for excellent supervision throughout the project. I have never before met a person with such passion and enthusiasm for his work. His knowledge has been invaluable for this project.

I was a guest at University of Pennsylvania in the Electrical and Systems Engineering Department from February 2011 till August 2011. I would like to show my gratitude to Professor George J. Pappas for making this stay possible, and for supervising me during the stay. I feel that the stay gave some new perspectives on my project, which have greatly affected the remainder of the project.

I would also like to thank MT-LAB, CISS, and the Distributed and Embedded Systems Unit for supporting my research.

Furthermore, I would like to thank my colleagues at both the Distributed and Embedded Systems Unit and Section for Automation and Control for creating a friendly environment.

Finally, I would like to thank my family and friends for supporting me throughout the past three years.

Abstract

The purpose of this thesis is to develop a method for verifying timed temporal properties of continuous dynamical systems, and to develop a method for verifying the safety of an interconnection of continuous systems. The methods must be scalable in the number of continuous variables and the verification procedures should be algorithmically synthesizable.

Autonomous control plays an important role in many safety-critical systems. This implies that a malfunction in the control system can have catastrophic consequences, e.g., in space applications where a design flaw can result in large economic losses. Furthermore, a malfunction in the control system of a surgical robot may cause death of patients. The previous examples involve complex systems that are required to operate according to complex specifications. The systems cannot be formally verified by modern verification techniques, due to the high complexity of both the dynamical system and the specification. Therefore, there is a need for methods capable of verifying complex specifications of complex systems.

The verification of high dimensional continuous dynamical systems is the key to verifying general systems. In this thesis, an abstraction approach is taken to the verification problem. A method is developed for abstracting continuous dynamical systems by timed automata. This method is based on subdividing the state space into cells by use of subdivisioning functions that are decreasing along the vector field. To allow the verification of timed temporal properties, the continuous systems are abstracted by timed automata; hence, the Lie derivatives of the functions subdividing the state space are also used to generate time information for the abstraction. An algorithm for generating the abstraction for polynomial vector fields is developed based on sum of squares programming. In addition, a necessary and sufficient condition is provided for identifying the subdivisioning functions that allow the generation of complete abstractions. A complete abstraction makes it possible to verify and falsify timed temporal properties of continuous dynamical systems, whereas a sound abstraction allows the verification of universally quantified timed temporal properties on positive normal form of continuous dynamical systems.

To allow safety verification of higher dimensional dynamical systems, a compositional safety verification technique is developed. It is shown that dual decomposition can be applied on the problem of generating barrier certificates, resulting in a compositional formulation of the safety verification problem. This makes the barrier certificate method applicable to the verification of high dimensional systems, but at the cost of conservatism in terms of constraining the barrier functions to be additively separable.

The developed abstraction method enables verification of timed specifications for continuous dynamical systems, which other methods are not capable of. In addition, the increased information in the abstraction in terms of time makes the method useful for,

e.g., improved safety verification. Finally, the compositional safety verification allows verification of high dimensional systems.

Synopsis

Formålet med denne afhandling er, at udvikle en metode til verificering af tidsspecifikationer for kontinuerte dynamiske systemer, og at udvikle en metode til sikkerhedsverifikation af kontinuerte systemer. Metoderne skal være skalerbare i antallet af kontinuerte variable, og verifikationsproceduren skal kunne udføres af algoritmisk.

Autonom regulering spiller en vigtig rolle i mange sikkerhedskritiske systemer. Dette indebærer, at en fejl i et reguleringssystem kan forårsage katastrofale konsekvenser, fx kan en designfejl i en rum mission forårsage et stort økonomisk tab. Ydermere kan en fejl i reguleringssystemet af en kirurgiskrobot medføre død for en patient. Disse eksempler er givet af komplekse systemer, der skal operere i overensstemmelse med deres specifikation. Systemerne kan ikke blive formelt verificeret med moderne verificeringsteknikker, grundet den høje kompleksitet af både system og specifikation. Derfor er der et behov for verifikationsmetoder, der er i stand til at verificere komplekse specifikationer for komplekse systemer.

Verifikation af kontinuerte dynamiske systemer af høj dimension er nøglen til at verificere generelle systemer. I denne afhandling genereres abstraktioner for at muliggøre verifikationen. En metode er udviklet til abstraktion af kontinuerte dynamiske systemer med tidsautomater. Metoden er baseret på en inddeling af tilstandsrummet ved brug af inddelingsfunktioner, der er aftagende i retning af vektorfeltet. For at muliggøre verifikation af tidsspecifikationer er de kontinuerte dynamiske systemer abstraherede med tidsautomater. Hermed benyttes den Lie afledede af inddelingsfunktionerne også til generering af tidsinformation i den abstrakte model. En algoritme til generering af abstraktionen er udviklet til polynomiske vektorfelter ved brug af summe af kvadrerede polynomier. Ydermere er en tilstrækkelig og nødvendig betingelse for inddelingsfunktionerne fundet, der fortæller hvilke funktioner, der kan benyttes til generering af fuldstændige abstraktioner. En fuldstændig abstraktion muliggør verificering of falsifikation af tids temporale specifikationer af kontinuerte dynamiske systemer, mens andre abstraktioner kun tillader verificering af universelt kvantificerede specifikationer.

For at muliggøre sikkerhedsverifikation af højereordenssystemer er en kompositionel metode udviklet. Det er vist at en dual dekomposition af problemet, der benyttes til generering af barriere certifikater, resulterer i en kompositionel formulering af problemet. Dette gør barriere certifikat metoden anvendelig til verificering af systemer af høj dimension, dog er metoden mere konservativ end den oprindelige metode, da klassen af tilladelige funktioner er mindre.

Den udviklede abstraktionsmetode muliggør verifikation af tidsspecifikationer for kontinuerte dynamiske systemer, som andre metoder ikke kan håndtere. Ydermere har metoden vist sig brugbar til verificering af sikkerhedsspecifikationer. Sidst muliggør metoden verificering af systemer af høj dimension.

1 | Introduction

This chapter provides a motivation for studying verification of continuous systems, gives an overview of related literature, states the research hypothesis, and outlines the remainder of the thesis.

1.1 Motivation

Every system is designed to have an intended behavior that fulfills its functional requirements, but most often it is not clear if a system actually always exhibits the intended behavior. Correct behavior is not crucial for all systems, as long as the systems most often or almost satisfy the specification. Today's systems are getting larger and more complex, have more elaborate requirements, and are expected to satisfy their specification. Furthermore, autonomous control is being integrated in safety-critical systems such as surgical robots, where an undesired behavior may cause mortality [MHDK11]. Therefore, there is a need for methods that guarantee the correctness of systems.

Formal verification is concerned with the generation of a proof of correctness for a system model with respect to an associated specification. The formal verification provides an answer to the following problem.

Problem 1*: Given a system model and a specification of allowed and required behavior. Determine if the possible behavior of the system complies with the specification.*

The verification problem is generally difficult to solve; in fact, it can only be solved for certain classes of systems and specifications [HKPV98, AHLP00, LPS00].

Formal verification has been successfully treated in the computer science community for software verification and program analysis, and more recently there has been a growing interest in the verification of control systems. This includes the verification of hybrid systems, cyber-physical systems, systems consisting of interacting agents, and autonomous systems operating in challenging environments [Alu11]. In particular, formal verification methods have been applied to robotic applications [DGH$^+$11, BBE$^+$07] and avionics [TMG01]. In this thesis, formal verification is applied to wind turbines, to show that a wind turbine does not exceed its load bound during an emergency shutdown procedure, and to design a controller that ensures the safety of a surgical robot, which compensates for the physiological motions of a patient during heart surgery.

It is only possible to verify a limited class of dynamical systems exactly [AHLP00]. Therefore, most methods for verifying dynamical systems rely on some degree of approx-

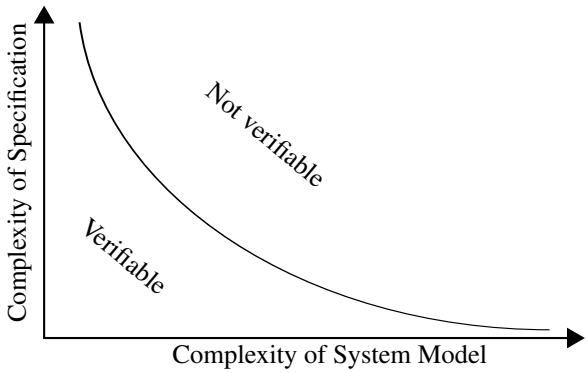


Figure 1.1: Graph of a line showing the frontier of system verification. It indicates that complex specifications can be formally verified for simple systems, and as the complexity of the systems increases, the expressiveness of the specification that can be verified decreases.

imation. The consequence of approximation is that a verification algorithm may give an inconclusive answer to Problem 1. For example, a method based on over-approximating the reachable set of a system can verify safety, but not falsify safety. This issue depends on both the specification and the considered system model.

The models used in the control of practical systems often have a large number of continuous variables. Therefore, the applicability of verification methods depends to a high extend on their scalability properties. This is a general issue that also applies to more traditional controller design methods. Methods based on linear matrix inequalities (LMIs) are very popular in the control community, due to their scalability properties, and even though the class of systems that can be treated with LMIs is restricted. When the LMI-framework is extended to sum of squares, scalability issues arise and the applicability decreases.

Recently, a lot of research effort has been put into the development of compositional methods in an attempt to alleviate the scalability issues. These methods allow the verification of a system by individual verification of subsystems, via composing the proofs for the subsystems into a proof for the interconnected system. However, it is difficult to decompose a continuous system into subsystems without making the subsequent analysis too conservative, as the interconnection of the subsystems is given by continuous signals.

Figure 1.1 depicts the trend of the tradeoff that is faced in system verification. The line indicates that it is possible to verify complex specifications of simple system models, and shows that only simple specifications can be verified for complex systems. Note that a measure of system complexity of a continuous system should include both dimension of the system and structure of the vector field.

The desire to move this frontier is the motivation for this work. The overall aim of this thesis is to move the frontier to *allow formal verification of more complex specifications for more complex system models*.

The verification of continuous dynamical systems is important for the verification of both continuous and hybrid systems, as this is the main source of complexity in the

verification procedure [GLZN09]. For continuous systems, the verification of simple properties such as stability may be difficult. Therefore, it is also relevant to extend the specifications that can be verified for a class of continuous systems.

Some verification methods are tailored to proving only one particular property of a system. These methods are often more elegant and less complex to solve than general verification methods. This underpins the important point that a verification method should be chosen based on the system class and the property that should be verified.

Safety verification of continuous systems is worth its own study, and has received a lot of attention. A system is safe if it cannot reach an unsafe subset of its state space from a set of initial states. One of the motivations for studying model predictive control is the inclusion of state constraints in the formulation [CB99], which can be expressed as a safety specification.

Although verification problems are interesting, the synthesis of provably correct systems is even more intriguing. However, the synthesis problem is more difficult than the verification. Therefore, the literature is more limited on this subject. The primary focus of this thesis is formal verification. In fact, only Paper F is concerned with synthesis of safe controllers.

This thesis addresses the following:

- Development of a verification method for continuous systems that allows the verification of timed temporal logic specifications. The method is based on abstracting continuous systems by timed automata.

- Development of a method for verifying the safety of high-dimensional continuous systems. It is based on the generation of barrier certificates and decomposing the verification problem into coupled subproblems.

- Development of a method for synthesizing safe controllers for continuous systems. The method is also based on the barrier certificate method.

1.2 State-of-the-Art and Background

The purpose of this section is to introduce work that is similar to the contributions of this thesis, to show how the developed methods fit into the state-of-the-art. The thesis contains contributions in relation to abstractions of continuous systems, safety verification by barrier certificates, and design of safe control systems. Therefore, the section is focused on these topics. This is not a thorough survey, but only a small selection of relevant methods. At first verification methods are described, and at the end controller design methods are provided.

Verification Methods

Verification methods for dynamical systems can be classified into direct and indirect methods. Direct verification methods accomplish the verification directly on the model of a considered system, whereas indirect methods consider a simplified or abstract model of the considered system in the verification. The verification method presented in Chapter 3 is an indirect method, as it is based on abstracting the considered system by a timed

automaton. Throughout the section, the literature is compared to the developed methods, and the abstraction developed in Chapter 3 is referred to as *TA-abstraction*.

The abstraction method developed in [TK04, Tiw08, TK02] is in the class of sign based abstractions described in [Tab09, p. 94].

The principle of the method is to generate an abstraction of a dynamical system by a finite automaton. The abstraction is generated by subdividing the state space into cells and letting each cell be abstracted by a location of a finite automaton. The subdivision of the state space is generated by level sets of multiple functions. A transition is added from a location to another location, if the two locations abstract neighboring cells in the state space, and the vector field points out of the boundary of the source cell into the destination cell. The existence of a transition is determined by calculating the Lie derivative of the function used to subdivide the state space with respect to the vector field. The abstract model is a finite automaton; hence, properties of it can be verified. By appropriately defining the behavior of the abstraction and the dynamical system, it can be shown that a sign based abstraction simulates the dynamical system, i.e., properties such as safety can be verified.

The computations involved in the generation of a sign based abstraction are relatively simple, as the solution to the differential equation is not used. For multi-affine systems (i.e., the degree of the vector field in any of the variables is less than or equal to one [KB08]) with a rectangular subdivision of the state space, the computations of the Lie derivative can be done efficiently [KB06]. However, there is no systematic way of choosing functions for subdividing the state space to generate a sign based abstraction. Instead, a heuristic is proposed for refining the abstraction, where additional functions are added to subdivide the state space. Roughly, if a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is used in the subdivision of the state space, then the Lie derivative along the vector field f is added to the set of functions subdividing the state space.

Although a sign based abstraction is sound, the calculated reachable set can be very inaccurate. This is clear from the following example.

Example 1. Consider a dynamical system, given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{1.1}$$

The subdivision of the state space is illustrated in Figure 1.2 by black lines, the vector field is illustrated by blue arrows, and the set of initial states is shaded gray.

By use of the sign based abstraction, the entire first quadrant is reachable, but actually only a block diagonal trace (cells within the bold black line in Figure 1.2) is reachable.

The example shows that the sign based abstraction generates a poor abstraction of the reachable set.

Remark 1*: The similarities between the sign based abstraction and the TA-abstraction is that level sets of multiple functions are used to subdivide the state space, and that the transitions in the abstract model are generated based on the Lie derivative of the function used to subdivide the state space with respect to the vector field. In fact, the admissible subdivisioning functions in the TA-abstraction is only a subset of the admissible subdivisioning functions of this abstraction, since the Lie derivative of the subdivisioning functions with respect to the vector field must be decreasing in a TA-abstraction (except at critical points).*

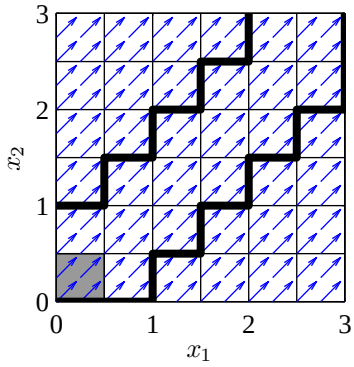


Figure 1.2: Phase plot of a system with constant derivatives. The system is initialized in the bottom left cell (gray). If an automaton is generated based on the Lie derivative of the boundary of cells, we can only conclude that the reachable set is a subset of the first quadrant.

Finally, we provide conditions for generating a complete abstraction, whereas this method only provides a heuristics for generating additional subdivisioning functions based on an initial collection of subdivisioning functions.

Example 1 was the inspiration for [MB08] that extends the sign based abstraction with time information, by abstracting the dynamical system by a timed automaton instead of an automaton. The addition of time information improves the accuracy of the approximation of the reachable set, compared to the sign based abstraction. By using the method in [MB08], the approximated reachable set of Example 1 is reduced from the entire first quadrant to the cells within the bold black line in Figure 1.2. Note that these cells are exactly the cells reached by the dynamical system.

The abstraction method presented in [MB08] is more complicated to generate than the sign based abstraction, as it requires the calculation of the maximum time for the solution trajectories to traverse a cell and minimum and maximum times for solutions to traverse the slices used to generate the abstraction. Therefore, the timed automaton abstracting an n dimensional dynamical system has $2n + 1$ clocks.

In this abstraction, the partition of the state space is not generated in accordance with the vector field. Therefore, the improvement of the abstraction is subtle for some dynamical systems. This issue is illustrated in the following example.

Example 2. Consider a dynamical system, given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.1 & 1 \\ -1 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{1.2}$$

The subdivision of the state space is illustrated in Figure 1.3 by black lines.

For the subdivision of the state space given in Figure 1.3, the minimum time that a solution can maintain in any slice (collection of cells along a coordinate axis) is zero time units. Furthermore, the maximum time for traversing a slice that includes the equilib-

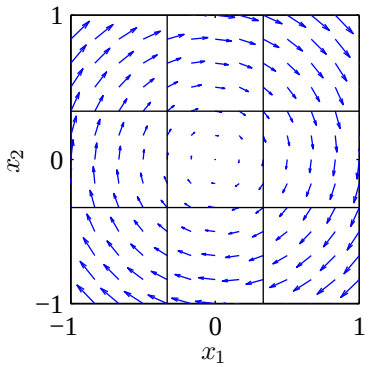


Figure 1.3: Phase plot for the system shown in (1.2) and a partition of its state space into 9 cells.

rium point is infinite. Therefore, the approximation of the reachable set is poor for this dynamical system.

Example 2 indicates that the addition of time in the abstraction can improve calculations of reachable sets, but a closer resemblance between the dynamical system and the abstraction is needed to describe the reachable set for dynamical systems.

An abstraction method preserving more information about the vector field was developed in [Bro99]. This method shows how a bisimulation of a hybrid system can be constructed. The abstraction is obtained based on a partition of the state space that is generated by tangential and transversal foliations. The method is based on the local existence of such abstractions, via the flow box theorem; hence, in general, the partition is not constructive. The method has similarities with reduction techniques for mechanical systems, which also relies on the identification of first integrals [Gol60].

Remark 2: Reduction methods for mechanical systems are also the outset of Paper C that shows how the TA-abstraction can be used to abstract mechanical systems, by use of reduction techniques.

The principle of the barrier certificate method presented in [PJP07] is to find a positive invariant set given as a sublevel set of a function (the level set is called a barrier) that includes the initial set, but excludes the unsafe set. If such a set can be found, then the system is safe.

The principle of the barrier certificate method is very simple, as the method is designed to verify only one property: safety. An advantage of the barrier certificate method is that barrier certificates can be easily computed by utilizing methods similar to searching for a Lyapunov function. The paper [PJP07], provides both a strict barrier certificate that is possibly nonconvex and difficult to compute, and a weak barrier certificate that is convex and can be computed. The method is extended in [Pra06] to allow discontinuous barrier certificates and bounded time analysis, and the dual problem to safety (reachability) is addressed in [PR05b]. Additionally, [PR07] shows how the framework can be used to verify temporal logic properties.

In [PR05a], it is shown that there exists a barrier certificate if and only if the safety

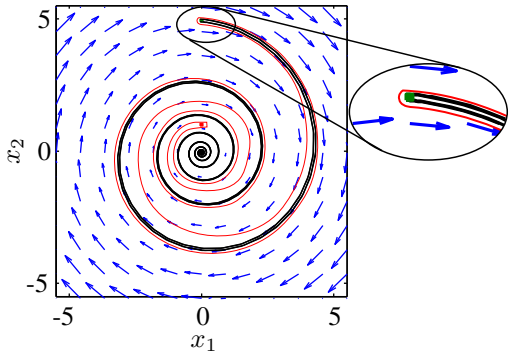


Figure 1.4: Illustration of the initial set (green box) and unsafe set (red box) together with some admissible system trajectories (block lines). The red line is the graph of a barrier certificate that proves the safety of the system.

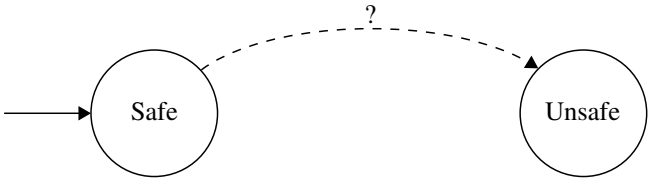


Figure 1.5: Automaton with two states: ”Safe” and ”Unsafe”. The existence of a barrier certificate determines if the is a transition between the two locations.

property holds, under the assumption of the existence of a decreasing function on the entire state space. Although a barrier function exists, it may be very complicated and therefore practically impossible to find. This is exemplified in the following example from Paper B.

Example 3. Consider the two-dimensional linear system given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.1 & 1 \\ -1 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{1.3}$$

with initial set $X_0 = [-0.05,\, 0.05] \times [4.95,\, 5.05]$ and unsafe set $X_{\mathrm{u}} = [-0.05,\, 0.05] \times [0.85,\, 0.95]$.

To verify the safety of this system, a very high degree barrier certificate must be identified. A barrier certificate is illustrated in Figure 1.4 by the red line, which is a level set or barrier separating the initial set and unsafe set. Note that the barrier certificate is generated by hand, as an attempt to solve the problem in SOSTOOLS failed [PPSP05].

The barrier certificate method can be seen as a very simple abstraction consisting of two locations, as depicted in Figure 1.5. One state has the predicate ”Safe” and the other state has the predicate ”Unsafe”. If one finds a barrier certificate, then there is no transition between the two locations; hence, the system is safe.

Remark 3: The barrier certificate is a special instance of the TA-abstraction, and this particular choice of subdivisioning function would result in an abstraction given by a timed automaton with two locations, where all solutions stay in the initial location for all times. The TA-abstraction allows the use of multiple subdivisioning functions; hence, these functions may be of lower degree, at the cost of the need for several level sets.

To allow the verification of a control system with an infinite state space, it is appropriate to generate finite state abstractions of the control system. In [Tab07] a control system is abstracted by a symbolic model that is bisimilar to the control system, but is finite. The abstraction method restricts the number of input symbols to be finite, and to obtain a finite model, the state space must be partitioned into a finite number of equivalence classes.

In [GP07] the notion of approximate (bi)simulation was introduced for continuous systems. The method is based on describing continuous systems by metric transition systems, to get a common framework for treating discrete and continuous systems. Having a metric on top of the transition system allows the use of approximate relations between systems. The distance between two systems may be difficult to compute, but it can be approximated by (bi)simulation functions that are similar to Lyapunov functions. The approximate bisimulation method is used in [GP06] to verify systems based on simulations. This is possible by associating functions to the system that determines how close simulated trajectories are if they are initialized within some distance. The generation of trajectories guaranteeing the verification for hybrid systems is presented in [JFA+07]. The work is extended to the verification of temporal properties of continuous systems in [FGP06]. Additionally, the method is extended to the controller synthesis problem for reachability and safety specifications [Gir12].

In [PLS00], the concept of hierarchically consistent control systems is introduced. This is a formalism for abstracting control systems by other simpler control systems, i.e., the abstraction is continuous. An abstraction of a control system must be able to produce the same output trajectories as the control system it abstracts (i.e. the abstraction is an over-approximation). However, the input to the two system need not be the same. This is a major difference between the hierarchical framework and model reduction, as the hierarchical framework allows the introduction of virtual inputs in the abstraction. The virtual inputs enable an abstraction to have much lower dimension that the original system. Hence, analyzing the abstract system is simpler than analyzing the original system.

There are a lot of direct methods for computing the reachable set of systems. This means that the reachable set is calculated without any abstraction.

In [TMBO03] verification methods based on reachability calculations are categorized into over-approximative and convergent approximation methods. All methods described until this point (including the method proposed in this thesis) are over-approximative methods. However, the method presented in [MBT05] is a convergent approximation method. In [MBT05], it is shown that the backwards reachable set from a target set (described as a sublevel set of a continuous function) is the zero sublevel of the viscosity solution of a time-dependent Hamilton–Jacobi–Isaacs partial differential equation. The solution is calculated on a grid of the state space, but the method is computationally demanding and can only be used for verification of low dimensional systems. The method has especially been applied to problems in avionics, including [TMG01], since it allows nonlinear vector fields. Since the method is based on game theoretic principles, it can be used for synthesizing controllers that guarantee some objective despite the presence of

disturbances [TLSS00]. A toolbox for finding the viscosity solution of a time-dependent Hamilton–Jacobi–Isaacs partial differential equation is developed in [MT05].

The principle of a series of methods for directly calculating the reachable set of a system, is to propagate the set of corner points of an initial set one time step, take the convex hull of the initial set and the one step successor, and enlarge this set to ensure that the approximation is an over-approximation. One of the crucial choices of these methods is the choice of class of sets for representing the reachable set, as it greatly influences the computation time. In [CK03] the reachability calculation used in the verification tool CheckMate is presented. The tool can handle switching in systems and is based on convex polyhedra. Polyhedral sets are also used in the tool PHAVer (Polyhedral Hybrid Automaton Verifyer) for reachability calculations of linear hybrid automata. A similar method based on zonotopes is presented in [Gir05], which gives improved computational times. In [Gir05], a particular emphasis is put on reachability calculations for uncertain linear systems, as the contribution of the uncertainty can be decoupled from the linear dynamics in the solution of the problem, eliminating the wrapping effect. In [ASB10] a method based on a combination of zonotopes and polyhedra is presented. Finally, the more recent tool SpaceEx combines polyhedra and support function to improve the scalability of the computations [FLGD$^+$11].

A collection of methods based on ellipsoids are also developed. These methods are based the computational methods presented in [KV97], and enables estimation of reachable sets using collections of ellipsoids, based on both under-approximation and over-approximation. The methods applies for varies classes of systems [KV00, KV07], and a toolbox called Ellipsoidal Toolbox has been developed for the computations [KV06].

Compositional Verification Methods

Compositional verification methods have been developed in an attempt to improve the scalability of the verification.

The principle of assume-guarantee reasoning presented in [KvdS10] for linear systems is to consider a system Σ_{P} given as an interconnection of N systems, i.e., $\Sigma_{\mathrm{P}} = \Sigma_{\mathrm{P}_1} || \dots || \Sigma_{\mathrm{P}_N}$. It is desired to verify if the system satisfies a specification that is also given as an interconnection of systems $\Sigma_{\mathrm{Q}} = \Sigma_{\mathrm{Q}_1} || \dots || \Sigma_{\mathrm{Q}_N}$. The verification is accomplished by checking if there exists a simulation relation of Σ_{P} by Σ_{Q}

$$\Sigma_{\mathrm{P}_1} || \dots || \Sigma_{\mathrm{P}_N} \preceq \Sigma_{\mathrm{Q}_1} || \dots || \Sigma_{\mathrm{Q}_N}, \tag{1.4}$$

where $\Sigma_1 \preceq \Sigma_2$ denotes that Σ_1 is simulated by Σ_2. If $\Sigma_1 \preceq \Sigma_2$ then the output trajectories of Σ_1 is a subset of the output trajectories of Σ_2.

The assume-guarantee framework can be used to do compositional reasoning, using the following property

$$\begin{cases} \Sigma_{P_1} \preceq \Sigma_{Q_1} \\ \Sigma_{P_2} \preceq \Sigma_{Q_2} \end{cases} \Rightarrow \Sigma_{P_1} || \Sigma_{P_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2}. \tag{1.5}$$

This property allows individual verification of subsystems. If each subsystem satisfies its subspecification, then the interconnected system satisfies the overall specification.

The following properties are used for assume-guarantee reasoning

$$\begin{cases} \Sigma_{P_1} \preceq \Sigma_{Q_1} \\ \Sigma_{Q_1} || \Sigma_{P_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2} \end{cases} \Rightarrow \Sigma_{P_1} || \Sigma_{P_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2} \tag{1.6}$$

$$\begin{cases} \Sigma_{P_1} || \Sigma_{Q_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2} \\ \Sigma_{Q_1} || \Sigma_{P_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2} \end{cases} \Rightarrow \Sigma_{P_1} || \Sigma_{P_2} \preceq \Sigma_{Q_1} || \Sigma_{Q_2}. \tag{1.7}$$

These properties allow each subsystem to be interconnected with the specification and then verified. The intuition of the assume-guarantee reasoning is to verify a subsystem, assuming that the remaining subsystems satisfy their subspecification. Note that assumptions about the dynamical systems are necessary to accomplish the assume-guarantee reasoning.

A method for compositional stability analysis is presented in [TPM09]. In this paper a framework is proposed for analyzing the stability of a system decomposed into subsystem. In addition to the stability analysis, the region of attraction can be determined in a compositional manner. The method is based on dual decomposition [DW61]; hence, the stability problem is decomposed into subproblems, which are coupled via a master problem with less variables. This implies that subsystems are not completely separated as in (1.5).

Remark 4*: The compositional stability method is similar to the compositional barrier certificate method presented in Paper D and Chapter 4. In both methods the compositional conditions can be modified to satisfy* (1.5)*, but this is considered to be too conservative in general. Specifically, this is accomplished by letting* $\alpha_i = 0$, $\beta_i = 0$, $\gamma_i = 0$ *for all* i *in Corollary 4.*

Controller Design Methods

Several of the presented verification methods are extended to allow the design of controllers. These methods are used to guarantee a specification by design; hence, eliminating the need for verification. An issue with the majority of these methods is that the models have a lot of discrete states, and numerous input symbols; hence, the generation of a controller is difficult.

The barrier certificate method is extended to the design of safe controllers in [WA07]. The extension is inspired by the extension of Lyapunov functions to control Lyapunov functions. To simultaneously finding a controller and a barrier certificate is in general very difficult; however, the paper provides a constructive method for finding a safe state feedback controller for a given control barrier certificate. This implies that a state feedback controller can be found analytically.

Remark 5*: This work was the outset for Paper F that provides a refined definition of a control barrier function, and allows disturbances in the controller design.*

The symbolic methods for verifying and comparing systems via (bi)simulation and approximate (bi)simulation are also extended to the design of controllers in [Tab09, PT09, JT11]. For this particular method the tool Pessoa is developed [MDT10]. Pessoa allows generation of correct-by-design controllers for dynamical systems, but due to the abstrac-

tion method, it is only possible to generate controllers for low dimensional systems with a few coarsely discretized control signals.

In [LTS99], a game-based approach to satisfying reachability specifications was proposed. This method is similar to verification framework [TMBO03] explained earlier. This framework can handle expressive classes of vector fields, but only applies to systems of low dimension. The idea of receding horizon control has also been extended in [WTM10] for the synthesis of receding horizon temporal logic planning.

In [TP03], it is shown that model checking LTL is decidable for controllable linear systems, by rewriting it to Brunovsky normal form. Then the discrete system represents a shift register with values determined by the control input. This system can easily be controlled to any desired state.

Similar to the use of invariant sets [Bla99], properties of the Lie derivative of the boundaries of rectangles is used to synthesize controllers in [BH06]. The idea of this method is to control the possible trajectories of a system, by partitioning the state space into rectangles and either control all trajectories to exit the each rectangle through a certain facet of make the rectangle positive invariant. The same idea is used in [KB08] for a triangulation of the state space to algorithmically generate controllers for linear systems that satisfies some temporal logic specification. The key idea of these methods is to control the vector field to be transversal to the facets, through the study of Lie derivatives.

1.3 Research Hypotheses

In the development of the verification methods in this thesis, ideas of the work presented in the previous sections are used to generate an abstraction that allows the verification of more expressive properties of continuous dynamical systems, and to develop a method for the safety verification of more complex (higher dimensional) systems. The philosophy used throughout the thesis is to only exploit the vector field - not solutions to differential equations or simulated trajectories. The following research hypotheses are tested in this work.

Research Hypothesis 1*: Subdividing the state space according to the vector field, allows the verification of TCTL properties of continuous systems via abstracting the system by a timed automaton.*

Research Hypothesis 2*: A decomposition of the safety verification problem allows the verification of higher dimensional systems.*

Hypothesis 1 is validated by generating a method that allows the verification of TCTL properties, showing how the state space should be partitioned in accordance with the vector field to generate an abstraction that preserves TCTL properties, and by showing that complete abstractions can be generated by providing examples.

Hypothesis 2 is validated by providing a compositional method for verifying the safety of continuous systems, and by showing that a system, for which the centralized verification method fails to verify, can be verified by the developed compositional method.

1.4 Outline of the Thesis

The thesis is divided into five chapters and an appendix of six papers, where the first chapter is this introduction. Chapter 2 presents a general introduction to formal verification, including logics and results on abstractions of system models. This provides some high level preliminaries for the work in the thesis. Chapter 3 and Chapter 4 present the content of the attached papers, where Chapter 3 presents the work in abstracting continuous systems by timed automata from Paper A-C and Chapter 4 presents the work in compositional safety verification from Paper D-E and synthesis of safe control systems from Paper F. These chapters are written to be self-contained and the paper should only be consulted for proofs, examples, or other details. Finally, Chapter 5 summarizes contributions, comprises conclusions and future work.

2 | Formal Verification

The purpose of this chapter is to provide a short background on the theory upon the work in this thesis relies.

Formal verification provides an answer to Problem 1 on page 1 for some definition of specification and model. A more precise formulation of the problem is provided by first presenting the logics used in specifications, and subsequently providing the overall principles of the verification. This gives an understanding of the class of systems that can be formally verified. Finally, the role of abstractions in the formal verification is outlined.

The difference between formal verification and verification by simulation, is that simulations usually cannot be utilized for proving that a system complies with its specification in every possible situation. However, formal verification provides a proof of correctness based on rigorous mathematical formulations. It is impossible to do exhaustive simulation for an infinite-state system. Therefore, verification using simulation is only possible if the considered system is finite and very simple, or extra measures are done in selecting the simulated trajectories based on knowledge about the system [FGP06].

2.1 Specifications

In the following, logics used to specify properties of systems are presented. The thesis is concerned with timed computation tree logic (TCTL), but linear temporal logic (LTL) is explained first, as this logic is used quite often [BK08, p. 235].

Definition 1 (Syntax of LTL)**.** An LTL formula over the set AP of atomic propositions is formed according to the following grammar

$$\psi ::= \text{true} \mid a \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \bigcirc \psi \mid \psi_1 U \psi_2,$$

where $a \in AP$.

In Definition 1, the symbol $\wedge$ means 'and', $\neg$ means 'not', $\bigcirc$ means 'next', and U reads until. Note that eventually ($\Diamond\psi$) and always ($\Box\psi$) can be expressed from the grammar as true$U\psi$ respectively $\neg\Diamond\neg\psi$.

The semantics of LTL is given in Figure 2.1 by examples, where a and b are atomic propositions.

The semantics of LTL can be given as a transition system, but this is not included due to its limited use in the thesis.

Computation tree logics (CTL) allow existential and universal quantification, and the extension timed computation tree logic (TCTL) allows the addition of time bounds in the

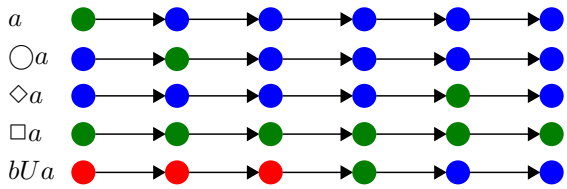


Figure 2.1: Semantics of LTL formulae [Alu]. Each line shows an execution of a system. The atomic proposition a is satisfied in the green circles, an atomic proposition b is satisfied in the red circles, and an arbitrary set of atomic propositions is satisfied in the blue circles.

logic [ACD90]. This makes it possible to express properties such as "before 5 time units" instead of "eventually". The syntax of TCTL is presented in the following.

Definition 2 (Syntax of TCTL)**.** The formulas ψ of TCTL are inductively defined as follows

$$\psi ::= \text{true} \mid a \mid g \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \exists\psi_1 U_J \psi_2 \mid \forall\psi_1 U_J \psi_2$$

where $a \in AP$, g is the set of atomic clock constraints, and $J \subseteq \mathbb{R}_{\geq 0}$ is an interval whose bounds are natural numbers.

Note that the quantifiers of CTL allow one to express that all traces should satisfy a property (this is implicitly assumed in LTL) or that only some trace should satisfy a property. Furthermore, the addition of time in TCTL allows one to specify that ψ_2 is satisfied within $t \in J \subseteq \mathbb{R}_{\geq 0}$ time units until which ψ_1 is satisfied, by the formula $\psi_1 U_J \psi_2$. For details in the semantics of TCTL, see [BK08]. A special fraction is considered in connection with the treatment of sound abstractions. This fragment is called universally quantified TCTL on positive normal form and is TCTL without existential quantification, and with negation only of atomic propositions.

Note that the expressiveness of CTL and LTL is incomparable, as both logics can express properties that the other logic cannot express.

2.2 Model Checking

There exist several methods for doing formal verification; however, only the principle of model checking is sketched in this chapter. Model checking is applicable for finite-state models, as it is based on a graph search of the system and specification. Therefore, its applicability to infinite-state systems depends on the ability to abstract the infinite-state systems by a finite-state model. In addition, only properties preserved by the abstraction of the infinite-state systems can be inferred on the original system based on the model checking of the finite-state abstraction. In the following, the procedure for generating a finite abstraction of a timed automaton is explained, to give an idea of the abstraction procedure.

A timed automaton is a hybrid system, where the continuous dynamics is given by clocks that increase at a constant rate of one, and the clocks can be reset to zero. To verify

a timed automaton, it is necessary to know how the values that the clocks evolve over time. This can be obtained from the region graph in Figure 2.2. Note that constraints on clocks are natural numbers, but clocks are real-valued; hence, the state space is infinite.

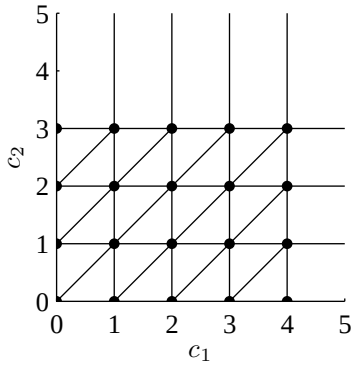


Figure 2.2: Region graph of a timed automaton, with two clocks c_1 and c_2.

The region graph in Figure 2.2 partitions the values of two clocks c_1 and c_2. The partition consists of three different types of elements: corner points, line segments, and triangles. The clocks are initialized at $(0, 0)$ and propagate from a corner point to a line segment, where the valuations can continue to propagate along the diagonal or be reset to zero. If clock c_2 is reset to zero, then all values of the clocks will be projected onto a line segment on the c_1-axis. All executions continue like this, since all clock valuations in a cell of the partition have the exact same abstract successors (both for time propagation and resets). Finally, by knowing the largest constant in the constraints, a finite region graph can be obtained, and CTL model checking can be applied on the region graph automaton. This implies that the abstraction preserves TCTL properties, making it possible to verify TCTL of timed automata. Tools exist for verifying both LTL and TCTL properties of timed automata [BDL04, BK08].

Abstractions for Verification

It is not always possible to generate a finite abstraction of a model. In [HKPV98], the decidable classes of hybrid automata are summarized, and the boundary of decidability is clarified. Furthermore, it is shown that by adding stopwatches to timed automata reachability is no longer decidable; however, other updates can be used for timed automata [BDFP00]. CTL and LTL model checking is also possible for o-minimal hybrid systems that are linear hybrid systems, where the system matrix is nilpotent, diagonalizable, or have purely imaginary eigenvalues with multiplicity one [AHLP00, LPS00].

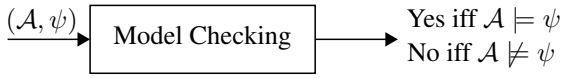


Figure 2.3: Principle of model checking if a system $\mathcal{A}$ satisfies a specification ψ.

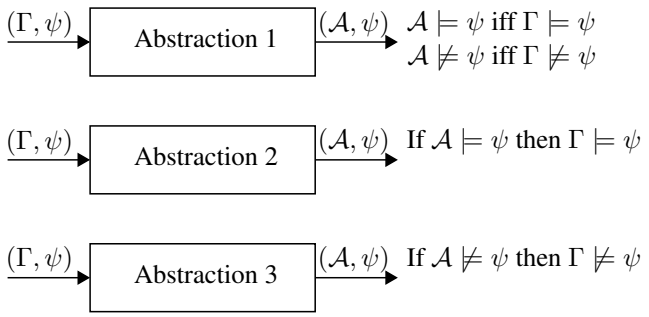


Figure 2.4: Three abstractions preserving different properties of the system Γ.

It is seen that the class of models for which a finite abstraction can be algorithmically generated, and used for model checking is limited. Therefore, one has to resort to abstractions that only preserve a selected fragment of the logic or abstractions that give conservative answers to the verification problem. This concept is shown in Figure 2.4, where three different abstractions are illustrated. Note that $\mathcal{A} \models \psi$ ($\mathcal{A} \not\models \psi$) means that $\mathcal{A}$ satisfies (does not satisfy) the formula ψ. The symbol $\models$ is also used when considering Γ, but the semantics of Γ and $\mathcal{A}$ are generated differently.

In Abstraction 1, a model Γ and associated specification ψ is abstracted by an abstract model $\mathcal{A}$ and associated specification ψ. It is seen that the abstraction preserves all properties of the chosen logic. Even though Abstraction 1 is the desired situation, it is not realistic for all system models and specifications. Therefore, other abstractions can be generated that enable the verification of desired properties (Abstraction 2 in Figure 2.4) or abstractions that enable the falsification of properties (Abstraction 3 in Figure 2.4). Let ψ express a safety property of system Γ. Abstraction 2 can be used to verify the safety, but if it fails to verify the safety then the system is not necessarily unsafe. Abstraction 3 can be used to falsify the safety, and if the falsification fails too, then we have an inconclusive answer to the verification problem, as the safety property has neither been proved nor disproved.

From Figure 2.3 and Figure 2.4, it is concluded that by generating an abstraction of some system by a timed automaton, it is possible to model check all TCTL properties that the abstraction preserves (similar applies for other logics). This implies that the abstraction of a continuous system by a timed automaton allows the model checking of the continuous system, to the extend that the abstraction preserves the properties.

Note that it is important for the applicability of an abstraction method in the latter two cases of Figure 2.4, to determine how well they approximate the system, as they are only sufficient conditions. A verification method that always returns false has the property of the second abstraction in Figure 2.4, but is of no use.

3 Abstracting Continuous Systems by Timed Automata

The purpose of this chapter is to present a method for abstracting continuous systems by timed automata. This abstraction method joins the sign based abstraction method with the introduction of time in [MB08], and is based on a subdivision of the state space generated by invariant sets. The method enables the following problem to be solved.

Problem 2: Given a continuous dynamical system and a TCTL specification over predicates given by subsets of the state space. Verify if the continuous system satisfies the specification.

Complete abstractions allow the verification and falsification of the problem, but sound abstractions only allow the verification of universally quantified TCTL specifications.

The chapter is an extract of the most important results from Paper A-C. Paper A and Paper B describe the abstraction framework for Morse-Smale vector fields. Paper C shows how the abstraction method relates to reduction of mechanical systems, and how a mechanical system can be abstracted by a timed automaton.

The chapter is organized as follows. Section 3.1 provides preliminary definitions related to dynamical systems and timed automata. In Section 3.2 the procedure for abstracting a dynamical system is presented including the subdivision of the state space and the generation of the timed automaton. Selected properties of the abstraction such as soundness and completeness are presented in Section 3.3, followed by algorithms for generating an abstraction in Section 3.4. In Section 3.5 the abstraction famework is extended to apply for mechanical systems, where the subdivision of the state space is generated based on reduction methods. Finally, Section 3.6 comprises conclusions.

Notation

The set $\{1, \dots, k\}$ is denoted by $\boldsymbol{k}$. B^A is the set of maps $A \to B$. The power set of A is denoted by 2^A. $|A|$ denotes the cardinality of the set A. We consider the Euclidean space $(\mathbb{R}^n, \langle , \rangle)$, where $\langle , \rangle$ is the standard scalar product. $\mathbb{N} = \{1, 2, \dots\}$ is the set of natural numbers, and $\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$ is the set of integers. $\mathfrak{X}^r(M)$ is the space of C^r vector fields. For a map $f : A \to B$, and a subset $C \subseteq A$, we define $f(C) \equiv \{f(x) |\ x \in C\}$. Whenever $f : X \to \mathbb{R}$ is a function and $a \in \mathbb{R}$, we write $f^{-1}(a)$ to shorten the notation of $f^{-1}(\{a\})$. Let $A \subseteq \mathbb{R}^n$, then $\mathrm{cl}(A)$ denotes the closure

of A. To say that a symmetric matrix M is positive definite (positive semidefinite), we write $M \succ 0$ ($M \succeq 0$).

3.1 Preliminaries

Dynamical systems and timed automata are detailed in this section. An emphasis is put on the importance of studying structurally stable vector fields to ensure robustness of the abstraction procedure. Furthermore, the notion of a solution trajectory of a timed automaton is introduced to ease the comparison of a dynamical system and a timed automaton.

Dynamical Systems

We denote a dynamical system by $\Gamma = (X, f)$, where $X \subseteq \mathbb{R}^n$ is the state space, and the dynamics is described by a system of ordinary differential equations $f : X \to \mathbb{R}^n$

$$\dot{x} = f(x). \tag{3.1}$$

The solution of (3.1), from an initial state $x_0 \in X_0 \subseteq X$ at time $t \geq 0$ is described by the flow function $\phi_\Gamma : [0, \epsilon] \times X \to X$, $\epsilon > 0$ satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{3.2}$$

for all $t \in [0, \epsilon]$ and with $\phi_\Gamma(0, x_0) = x_0$. The reachable set is defined as follows.

Definition 3 (Reachable set of Dynamical System)**.** The reachable set of a dynamical system Γ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1, t_2]$ is

$$\phi_\Gamma([t_1, t_2], X_0). \tag{3.3}$$

Considered Classes of Systems

We consider only Morse-Smale vector fields in this work, and develop algorithms for linear and polynomial systems.

A linear system $\Gamma = (X, f)$ is a system with a linear vector field f, i.e., given by

$$\dot{x} = Ax, \tag{3.4}$$

where A is an $n \times n$ non-singular matrix. Matrix A must be non-singular, to ensure that the system is hyperbolic. If the vector field f is a polynomial map, then the system $\Gamma = (X, f)$ is said to be a polynomial system. We only consider polynomials with real-valued variables, and for $n \geq 1$ we denote the polynomial ring $\mathbb{R}[x_1, \ldots, x_n]$ by $\mathbb{R}[\underline{x}]$. In addition, a map $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be polynomial if its coordinate functions are polynomials, i.e., $f_i \in \mathbb{R}[\underline{x}]$ for $i = 1, \ldots, m$.

The class of systems considered in this work is Morse-Smale systems.

Definition 4 (Morse-Smale System [Wis05])**.** A smooth vector field $X \in \mathfrak{X}^r(M)$ is called a Morse-Smale system (or field) provided it satisfies the following conditions:

1. X has a finite number of singular points, say $\beta_1, \ldots, \beta_k$, each hyperbolic. A hyperbolic singular point is a singular point such that in local coordinates the matrix of partial derivatives of X has eigenvalues with nonzero real parts.
2. X has a finite number of closed orbits (periodic solutions), say $\beta_{k+1}, \ldots, \beta_n$, each hyperbolic.
3. For any $p \in M$, $\alpha(p) = \beta_i$ and $\omega(p) = \beta_j$ for some i and j.
4. Let $\Omega(X)$ be the nonwandering[1] points for X, then $\Omega(X) = \{\beta_1, \ldots, \beta_N\}$.
5. The stable and unstable manifolds associated with the β_i have transversal intersection.

Two important restrictions of Morse-Smale systems are exploited in this work. First, the vector field has a finite number of singular points, each hyperbolic. This property allows a linear system to be split up into a stable and an unstable subsystem, which can be analyzed separately. Second, the stable and unstable manifolds associated with a singular point have transversal intersection. This restriction is necessary to obtain a structurally stable vector field.

Definition 5 (Structurally Stable Vector Field [JdM80])**.** A vector field $\xi \in \mathfrak{X}^r(M)$ is structurally stable if there exists a neighborhood V of ξ in $\mathfrak{X}^r(M)$ such that every $\eta \in V$ is topologically equivalent to ξ.

A vector field is structurally stable if its quantitative behavior does not change after the vector field has been slightly perturbed. To stress the importance of structural stability, an example from Paper A on page 83 is provided in the following.

Example 4. Consider the linear system with purely complex eigenvalues $\{-i, i\}$

$$\dot{x} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} x. \tag{3.5}$$

Graphs of two trajectories of the system are shown in the left subplot of Figure 3.1. By slightly perturbing the system (This perturbation is given by a smooth map, see Theorem 2.1 in [Hir76]), the real part of the eigenvalues of the system may become positive (trajectory in the middle subplot) or negative (trajectory in the right subplot). As a consequence, it is no longer possible to describe the solution trajectories of the middle or right subplot by a continuous deformation (homeomorphism) of solution trajectories of the left subplot. Therefore, the system given by (3.5) is *not* structurally stable.

In relation to numerical simulation of systems that are not structurally stable, even the smallest rounding error in the representation of the system may significantly alter its behavior.

Morse-Smale systems are dense in systems of less than or equal to two dimensions, see Theorem 8 and Theorem 9 in Paper A. In general, there are no methods for checking if a vector field is Morse-Smale; however, for a system of dimension greater than two, if

[1] We say that $p \in M$ is a wandering point for X if there exists a neighborhood V of p and a number t_0 such that $\phi_X(t, V) \cap V = \emptyset$ for $|t| > t_0$.

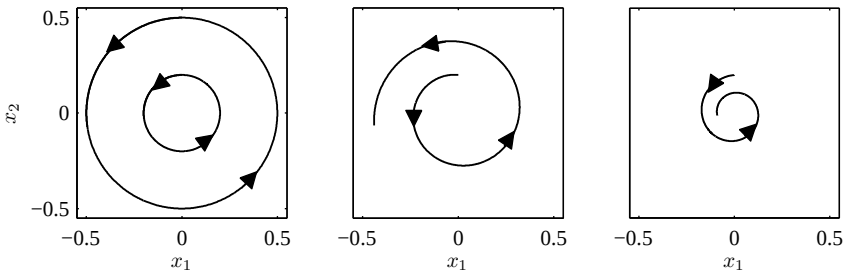


Figure 3.1: Trajectories of three dynamical systems.

there is a Lyapunov function then it can be approximated arbitrary closely by a Morse-Smale vector field [Mey68]. For linear systems, it is necessary and sufficient to check if all eigenvalues are hyperbolic.

Timed Automata

We use the notation of [AD94] in the definition of a timed automaton, and let $\Psi(C)$ be a set of diagonal-free clock constraints for a set of clocks C. This set contains all invariants and guards of the timed automaton, and is described by the following grammar

$$\psi ::= c \bowtie k | \psi_1 \wedge \psi_2, \tag{3.6a}$$

where

$$c \in C,\ k \in \mathbb{R}_{\geq 0},\ \text{and}\ \bowtie \in \{\leq, <, =, >, \geq\}. \tag{3.6b}$$

To make a clear distinction between syntax and semantics, the elements of $\bowtie$ are bold to indicate that they are syntactic operations. The semantics of the grammar is presented after the definition of a timed automaton.

Definition 6 (Timed Automaton)**.** A timed automaton $\mathcal{A}$ is a tuple $(E, E_0, C, \Sigma, I, \Delta)$, where

- E is a finite set of locations, and $E_0 \subseteq E$ is the set of initial locations;
- C is a finite set of clocks;
- Σ is the input alphabet;
- $I : E \to \Psi(C)$ assigns invariants to locations;
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations. A transition relation is a tuple $(e, G_{e\to e'}, \sigma, R_{e\to e'}, e')$ assigning an edge between two locations, where e is the source location and e' is the destination location. $G_{e\to e'} \in \Psi(C)$ is the set of guards, σ is a symbol in the alphabet Σ, and $R_{e\to e'} \subseteq C$ is a subset of clocks.

The semantics of a timed automaton is defined in the following.

Definition 7 (Clock Valuation)**.** A clock valuation on a set of clocks C is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation v_0 is given by $v_0(c) = 0$ for all $c \in C$. For a valuation v, a scalar $d \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + d$ and $v[R]$ are defined as

$$(v + d)(c) = v(c) + d, \tag{3.7a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{3.7b}$$

It is seen that (3.7a) is used to progress time and (3.7b) is used to reset the clocks in the set R to zero.

We denote the set of maps $v : C \to \mathbb{R}_{\geq 0}$ by $\mathbb{R}^C_{\geq 0}$. This notation indicates that we identify a valuation v with C-tuples of nonnegative reals in $\mathbb{R}^{|C|}_{\geq 0}$, where $|C|$ is the number of elements in C. We impose the Euclidean topology on $\mathbb{R}^C_{\geq 0}$.

Definition 8 (Semantics of Clock Constraint)**.** A clock constraint in $\Psi(C)$ is a set of clock valuations $\{v : C \to \mathbb{R}_{\geq 0}\}$ given by

$$[\![c \bowtie k]\!] = \{v : C \to \mathbb{R}_{\geq 0} |\ v(c) \bowtie k\} \tag{3.8a}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]. \tag{3.8b}$$

For convenience, we denote $v \in [\![\psi]\!]$ by $v \models \psi$ and denote the transition (e, v, σ, e', v') by $(e, v) \xrightarrow{\sigma} (e', v')$.

Definition 9 (Semantics of Timed Automaton)**.** The semantics of a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ is the transition system $[\![\mathcal{A}]\!] = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_\mathrm{s} \cup T_\mathrm{d})$, where S is the set of states

$$S = \{(e, v) \in E \times \mathbb{R}^C_{\geq 0} |\ v \models I(e)\},$$

$S_0 \subseteq S$ is the set of initial states

$$S_0 = \{(e, v) \in E_0 \times \mathbb{R}^C_{\geq 0} |\ v = v_0\}.$$

Note that $E \times \mathbb{R}^C_{\geq 0}$ induces subspace topology on S.

$T_\mathrm{s} \cup T_\mathrm{d}$ is the union of the following sets of transitions

$$\begin{aligned} T_\mathrm{s} = \{(e, v) \xrightarrow{\sigma} (e', v') |\ &\exists (e, G_{e\to e'}, \sigma, R_{e\to e'}, e') \in \Delta \\ &\text{such that } v \models G_{e\to e'} \text{ and } v' = v[R_{e\to e'}]\}, \\ T_\mathrm{d} = \{(e, v) \xrightarrow{d} (e, v + d) |\ &\forall d' \in [0, d] : v + d' \models I(e)\}. \end{aligned}$$

Hence, the semantics of a timed automaton is a transition system that comprises an infinite number of states: product of E and $\mathbb{R}^C_{\geq 0}$ and two types of transitions: the transition set T_s between discrete states with possibly a reset of clocks belonging to a subset $R_{e\to e'}$, and the transition set T_d that corresponds to time passing within the invariant $I(e)$.

In the following, we define an analog to the solution of a dynamical system for a timed automaton.

Definition 10 (Run of Timed Automaton)**.** A run of a timed automaton $\mathcal{A}$ is a possibly infinite sequence of alternations between time steps and discrete steps of the following form

$$(e_0, v_0) \xrightarrow{d_1} (e_0, v_1) \xrightarrow{\sigma_1} (e_1, v_2) \xrightarrow{d_2} \dots, \tag{3.9}$$

where $d_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$.

By forcing alternation of time and discrete steps in Definition 10, the time step d_i is the maximal time step between the discrete steps σ_{i-1} and σ_i.

To compare solution trajectories of a dynamical system with runs of a timed automaton, we define the continuous behavior of a timed automaton, in terms of a trajectory.

Trajectory of a Timed Automaton

A vital object for studying the behavior of any dynamical system is its trajectory. Therefore, we define a trajectory of a timed automaton. At the outset, we introduce a concept of a time domain.

In the following, we denote sets of the form $\{a, \dots\}$ with $a \in \mathbb{Z}_{\geq 0}$ as $\{a, \dots, \infty\}$. Let $k \in \mathbb{N} \cup \{\infty\}$; a subset $\mathcal{T}_k \subset \mathbb{Z}_{\geq 0} \times \mathbb{R}_{\geq 0}$ with disjoint (union) topology will be called a time domain if there exists an increasing sequence $\{t_i\}_{i \in \{0,\dots,k\}}$ in $\mathbb{R}_{\geq 0} \cup \{\infty\}$ such that

$$\mathcal{T}_k = \bigcup_{i \in \{1,\dots,k\}} \{i\} \times T_i,$$

where

$$T_i = \begin{cases} [t_{i-1}, t_i] & \text{if } t_i < \infty \\ [t_{i-1}, \infty[& \text{if } t_i = \infty. \end{cases}$$

Note that $T_i = [t_{i-1}, t_i]$ for all i if $k = \infty$. We say that the time domain is infinite if $k = \infty$ or $t_k = \infty$. The sequence $\{t_i\}_{i \in \{0,\dots,k\}}$ corresponding to a time domain will be called a switching sequence.

We define two projections $\pi_1 : E \times \mathbb{R}^C_{\geq 0} \to E$ and $\pi_2 : E \times \mathbb{R}^C_{\geq 0} \to \mathbb{R}^C_{\geq 0}$ by $\pi_1(e, v) = e$ and $\pi_2(e, v) = v$.

Definition 11 (Trajectory)**.** A trajectory of the timed automaton $\mathcal{A}$ is a pair $(\mathcal{T}_k, \gamma)$ where $k \in \mathbb{N} \cup \{\infty\}$ is fixed, and

- $\mathcal{T}_k \subset \mathbb{Z}_{\geq 0} \times \mathbb{R}_{\geq 0}$ is a time domain with corresponding sequence $\{t_i\}_{i \in \{0,\dots,k\}}$,
- $\gamma : \mathcal{T}_k \to S$ and recall that S is the (topological) space of joint continuous and discrete states; see Definition 9. The map γ satisfies:
  1. For each $i \in \{1, \dots, k-1\}$, there exists $\sigma \in \Sigma$ such that
  $$\gamma(i, t_i) \xrightarrow{\sigma} \gamma(i+1, t_i) \in T_{\mathrm{s}}.$$
  2. Let $\mathbf{0}$ be a vector of zeros and $\mathbf{1}$ be a vector of ones in $\mathbb{R}^{|C|}$, and recall that we identify a valuation $v \in \mathbb{R}^C_{\geq 0}$ with C-tuples of nonnegative reals in $\mathbb{R}^{|C|}_{\geq 0}$. For each $i \in \{1, \dots, k\}$
  $$\pi_2(\gamma(i, t_{i-1} + d)) = \pi_2(\gamma(i, t_{i-1})) + d\mathbf{1} \;\; \forall d \in \begin{cases} [0,\, t_i - t_{i-1}] & \text{if } t_i < \infty \\ [0,\, \infty[& \text{if } t_i = \infty \end{cases}$$

where $\pi_2(\gamma(i, t_{i-1} + d)) \in [\![I(\pi_1(\gamma(i, t_i)))]\!]$ and $\pi_2(\gamma(1, t_0)) = \mathbf{0}$. (Item 2 ensures that the time derivative of the valuation of each clock is one, between the discrete transitions.)

Note that γ is continuous by construction. Recall the definition of v_0 from Definition 7. A trajectory at (e, v_0) (with $v_0 \models I(e)$) is a trajectory $(\mathcal{T}_k, \gamma)$ with $\gamma(1, t_0) = (e, v_0)$.

Figure 3.2 shows the graph of a trajectory of a timed automaton with two clocks from Example 15 in Paper B. The dashed red lines and the blue lines illustrate valuations of the two clocks.

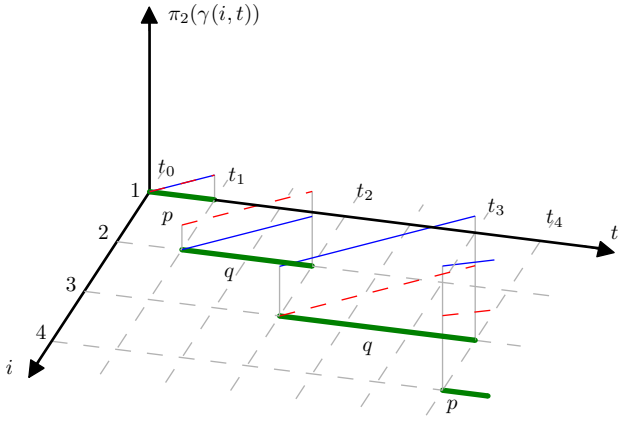


Figure 3.2: Trajectory of a timed automaton.

We define a discrete counterpart of the flow map.

Definition 12 (Flow Map of Timed Automaton)**.** The flow map of a timed automaton $\mathcal{A}$ is a multivalued map

$$\phi_{\mathcal{A}} : \mathbb{R}_{\geq 0} \times S_0 \to 2^S,$$

defined by $(e', v') \in \phi_{\mathcal{A}}(t; e, v_0)$ if and only if there exists a trajectory $(\mathcal{T}_k, \gamma)$ at (e, v_0) such that $t = t_k - t_0$ and $(e', v') = \gamma(k, t_k)$.

It will be instrumental to define a discrete flow map $\Phi_{\mathcal{A}} : \mathbb{R}_{\geq 0} \times E_0 \to 2^E$, which forgets the valuation of the clocks

$$\Phi_{\mathcal{A}}(t, e) = \pi_1 \circ \phi_{\mathcal{A}}(t; e, v_0). \tag{3.10}$$

In other words, $\Phi_{\mathcal{A}}$ is defined by: $e' \in \Phi_{\mathcal{A}}(t, e)$ if and only if there exists a run (3.9) of $[\![\mathcal{A}]\!]$ initialized in (e, v_0) that reaches the location e' at time $t = \sum_i d_i$.

The reachable set of a timed automaton is defined as follows.

Definition 13 (Reachable set of Timed Automaton)**.** The reachable locations of a timed automaton $\mathcal{A}$ from a set of initial locations $E_0 \subseteq E$ on the time interval $[t_1, t_2]$ is defined as

$$\Phi_{\mathcal{A}}([t_1, t_2], E_0) \equiv \bigcup_{(t,e) \in [t_1, t_2] \times E_0} \Phi_{\mathcal{A}}(t, e). \tag{3.11}$$

3.2 Abstractions of Dynamical Systems

The definition of a trajectory of a timed automaton enables a comparison between solution trajectories of dynamical systems and runs of timed automata. The comparison is accomplished in this section, by defining a so-called abstraction function, which associates subsets of the state space to locations of a timed automaton. Via the use of the abstraction function, we define sound and complete abstractions. Then a proposed subdivision of the state space is presented, based on the use of positively invariant sets. Finally, we show how a timed automaton can be generated from the subdivision of the state space.

We develop a concept of an abstraction of the dynamical system Γ. It consists of a finite number of sets $E \equiv \{e_\lambda |\ \lambda \in \Lambda \subseteq \mathbb{N}\}$, called cells. The cells cover the state space X

$$X = \bigcup_{\lambda \in \Lambda} e_\lambda.$$

To the subdivision E, we associate an abstraction function, which to each point in the state space associates the cells that this point belongs to.

Definition 14 (Abstraction Function)**.** Let $E \equiv \{e_\lambda |\ \lambda \in \Lambda \subseteq \mathbb{N}\}$ be a finite subdivision of the state space $X \subseteq \mathbb{R}^n$. An abstraction function for E is the multivalued function $\alpha_E : X \to 2^E$ defined by

$$\alpha_E(x) \equiv \{e \in E |\ x \in e\}. \tag{3.12}$$

For a given dynamical system Γ, we want to simultaneously devise a subdivision E of the state space X and create a timed automaton $\mathcal{A}$ with locations E such that

1. the abstraction is **sound** on an interval $[t_1, t_2]$:

$$\alpha_E \circ \phi_\Gamma(t, X_0) \subseteq \Phi_\mathcal{A}(t, \alpha_E(X_0)), \text{ for all } t \in [t_1, t_2]$$

2. the abstraction is **complete** on an interval $[t_1, t_2]$:

$$\alpha_E \circ \phi_\Gamma(t, X_0) = \Phi_\mathcal{A}(t, \alpha_E(X_0)) \text{ for all } t \in [t_1, t_2].$$

If a sound abstraction $\mathcal{A}$ is safe then Γ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$. Soundness is close to the notion of simulation; however, by soundness we relate different categories of models. Figure 3.3 illustrates the reachable set of a dynamical system, along with reachable sets of a sound abstraction (left) and a complete abstraction (right).

Remark 6: Note that the reachable abstract states of the timed automaton and the dynamical system are compared for every t *in the previous definition.*

In the verification of a system using an abstraction technique, it is paramount to know which properties we can infer about the system via a verification of the abstraction, as explained in Chapter 2. This is clarified in the following for sound and complete abstractions.

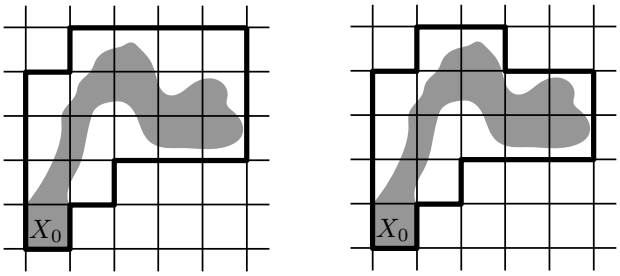


Figure 3.3: Reachable set of a dynamical system (shaded area), and reachable sets of automata (cells within bold lines). In the left figure, the reachable set of the automaton includes more cells than the ones reached by the dynamical system, i.e., the abstraction is sound. In the right figure, the reachable set of the automaton includes only the cells that are reached by the dynamical system, i.e., the abstraction is complete.

Proposition 1*: Let the timed automaton $\mathcal{A}$ be a sound abstraction of the dynamical system Γ and let ψ be a universally quantified TCTL formulae in positive normal form. Then $\mathcal{A} \models \psi$ implies that $\Gamma \models \psi$.*

Proof. The proof shows that the language of Γ is included in the language of $\mathcal{A}$. Then by Chapter 7 in [BK08], $\mathcal{A} \models \psi$ implies that $\Gamma \models \psi$, where ψ is a universally quantified TCTL formula in positive normal form.

Let the language of Γ be

$$L_\Gamma \equiv \bigcup_{x_0 \in X_0} (\tau, \sigma), \tag{3.13}$$

where $\tau = \tau_1, \tau_2, \ldots$ is an increasing sequence with $\tau_i \in \mathbb{R}_{\geq 0}$ and $\sigma = \sigma_0 \sigma_1 \ldots$ is an infinite word over 2^E with $\sigma_i = \alpha_E \circ \phi_\Gamma(\tau_i, x_0)$. The language of $\mathcal{A}$ is defined from σ and τ as

$$(\tau, \sigma) \in L_\mathcal{A}$$

if there exists a trajectory γ of $\mathcal{A}$ such that

$$\sigma_i = \bigcup_{k \in \mathbb{Z}_{\geq 0}} \pi_2 \circ \gamma(k, \tau_i).$$

Then $L_\Gamma \subseteq L_\mathcal{A}$, as $\alpha_E \circ \phi_\Gamma(t, X_0) \subseteq \Phi_\mathcal{A}(t, \alpha_E(X_0))$ for each t. Recall that $\phi_\mathcal{A}$ and subsequently $\Phi_\mathcal{A}$ are generated from the trajectory γ as shown in Definition 12. □

Proposition 2*: Let the timed automaton $\mathcal{A}$ be a complete abstraction of the dynamical system Γ. Then $\mathcal{A}$ and Γ satisfy the same TCTL formulae.*

Subdividing the State Space

This subsection presents the method used for subdividing the state space by functions, but not how the functions should be chosen to obtain. This is explained in Section 3.3 and

Section 3.4 provides algorithms for their generation. The subdivision of the state space is generated by intersecting sublevel sets of functions, and has two components: slices and cells. A slice is a sublevel set of one subdivisioning function, whereas a cell is a connected component of the intersection of sublevel sets of more subdivisioning functions.

The definition of a subdivision is motivated by the definition of a complex in algebraic topology [Bre93].

Definition 15 (Subdivision)**.** Let Λ be an index set, and $K = \{P_i\}_{i\in\Lambda}$ be a family of subsets in a Euclidean space $\mathbb{E}$. We let $|K| = \cup_{i\in\Lambda} P_i$ with the subspace topology inherited from $\mathbb{E}$. We call K a *subdivision* of a subset Y of $\mathbb{E}$, if

1. $\mathrm{int}(P) \neq \emptyset$, for all $P \in K$,
2. $P \cap P^{'}$ belongs to the boundary of P and $P^{'}$ for all $P, P^{'} \in K$,
3. each point of $|K|$ has a neighborhood intersecting only finitely many elements of K,
4. $|K| = Y$.

We define a slice as the set-difference of positively invariant sets.

Definition 16 (Slice)**.** A nonempty set S is a slice if there exist two open sets A_1 and A_2 such that

1. A_1 is a proper subset of A_2,
2. A_1 and A_2 are positively invariant, and
3. $S = \mathrm{cl}(A_2 \backslash A_1)$.

Since A_1 and A_2 are positively invariant sets, a trajectory initialized in S can propagate to A_1, but no solution initialized in A_1 can propagate to S. This implies that, via these invariants, we can study the possible trajectories of a dynamical system. We will adopt the convention that $\emptyset$ is a positively invariant set of any dynamical systems.

To devise a subdivision of a state space, we need to define collections of slices, called slice-families.

Definition 17 (Slice-Family)**.** Let $k \in \mathbb{N}$ and

$$A_0 \subset A_1 \subset \cdots \subset A_k$$

be a collection of positive invariant sets of a dynamical system $\Gamma = (X, f)$ with $X \subseteq A_k$ and $A_0 = \emptyset$. We say that the collection

$$\mathcal{S} \equiv \{S_i = \mathrm{cl}(A_i \backslash A_{i-1}) |\ i \in \boldsymbol{k}\}$$

is a slice-family generated by the sets $\{A_i |\ i \in \boldsymbol{k}\}$ or just a slice-family.

We associate a function to each slice-family $\mathcal{S}$ to provide a simple way of describing the boundary of a slice. Such a function is called a subdivisioning function.

Definition 18 (Subdivisioning Function)**.** Let $\Gamma = (X, f)$ be a dynamical system, let $\mathrm{Cr}(f)$ denote the set of critical points of f, and let $\mathcal{S}$ be a slice-family generated by the sets $\{A_i |\ i \in \boldsymbol{k}\}$. A continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \mathrm{Cr}(f)$ is a subdivisioning function for $\mathcal{S}$ if there is a sequence

$$a_0 < \ldots < a_k, \quad a_i \in \mathbb{R} \cup \{-\infty, \infty\},$$

where whenever $a_i \in \mathbb{R}$, it is a regular value of φ such that

$$\mathrm{cl}(A_i) = \varphi^{-1}([a_{i-1}, a_i]). \tag{3.14}$$

Remark that by regular level set theorem, for $a_i \in \mathbb{R}$, the boundary $\varphi^{-1}(a_i)$ of A_i is a smooth embedded submanifold of $\mathbb{R}^n$ of co-dimension 1 [Tu08].

We will create cells that cover the entire state space, by intersecting slices. To ensure robustness of the subdivision, it is important that the slices intersect transversally. The robustness of a transversal intersection is readily seen from the definition of transversal intersection [Hir76].

Definition 19 (Transversal Intersection)**.** Suppose that N_1 and N_2 are embedded submanifolds of M. We say that N_1 intersects N_2 transversally if, whenever $p \in N_1 \cap N_2$, we have $T_p(N_1) + T_p(N_2) = T_p(M)$. (The sum is not direct, just the set of sums of vectors, one from each of the two subspaces of the tangent space $T_p(M)$.)

The definition states that N_1 and N_2 are transversal if the tangent vectors to N_1 and N_2 span the entire space at each point of intersection. Hence, this transversality condition can be tested algorithmically.

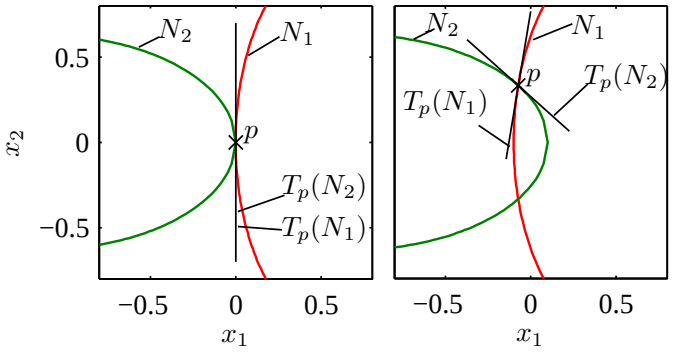


Figure 3.4: The left subplot shows an intersection that is not transversal; whereas, the right subplot shows a transversal intersection of two level sets.

The left subplot of Figure 3.4 illustrates level sets of two subdivisioning functions (hence two embedded submanifolds of $\mathbb{R}^2$). They intersect at the point p, and their tangents (black lines) generate a one dimensional subspace. This implies that their tangent vectors only span one dimension at p, i.e., $T_p(N_1) + T_p(N_2) \neq T_p(M)$. Therefore, this intersection is not transversal. Note that there exists an arbitrary small perturbation such that the intersection of the two level sets will be empty (this perturbation is given by a smooth map; see Theorem 2.1 in [Hir76]). Therefore, this subdivision is not robust.

In the right subplot of Figure 3.4, two level sets intersecting at point p are illustrated. Their tangent vectors (black lines) span $\mathbb{R}^2$, i.e., the level sets intersect transversally. Note that two manifolds that do not intersect are also transversal.

We define transversal intersection of slices as follows.

Definition 20 (Transversal Intersection of Slices)**.** We say that the slices S_1 and S_2 intersect each other transversally and write

$$S_1 \pitchfork S_2 = S_1 \cap S_2 \tag{3.15}$$

if their boundaries, $\text{bd}(S_1)$ and $\text{bd}(S_2)$, intersect each other transversally.

Cells are generated via intersecting slices.

Definition 21 (Extended Cell)**.** Let $\mathcal{S} = \{\mathcal{S}^i |\ i \in \boldsymbol{k}\}$ be a collection of k slice-families and let

$$\mathcal{G}(\mathcal{S}) \equiv \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\} \subset \mathbb{N}^k.$$

Denote the j^{th} slice in $\mathcal{S}^i$ by S^i_j and let $g \in \mathcal{G}(\mathcal{S})$. Then

$$e_{\text{ex},g} \equiv \pitchfork_{i=1}^{k}\ S^i_{g_i}, \tag{3.16}$$

where g_i is the i^{th} component of the vector g. Any nonempty set $e_{\text{ex},g}$ is called an extended cell of $\mathcal{S}$.

The cells in (3.16) are called extended cells, since the transversal intersection of slices may form multiple disjoint sets in the state space. It is desired to have cells, which are connected.

Definition 22 (Cell)**.** Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of k slice-families. A cell $e_{(g,h)}$ of $\mathcal{S}$ is a connected component of an extended cell of $\mathcal{S}$

$$\bigcup_h e_{(g,h)} = e_{\text{ex},g}, \text{ where} \tag{3.17a}$$

$$e_{(g,h)} \cap e_{(g,h')} = \emptyset \qquad \forall h \neq h'. \tag{3.17b}$$

There exist algorithms for determining the number of connected components of semi-algebraic sets given by polynomials, and providing semi-algebraic descriptions of the connected components. The number of connected components is given by the 0^{th} Betti number [Bre93]. An algorithm is presented in [BPR98] that takes as input a family of polynomials $\{P_1, \ldots, P_s\} \subset \mathbb{R}[x_1, \ldots, x_k]$ whose degrees are at most d, and outputs a semi-algebraic description of each connected component. The complexity of that algorithm is bounded by $s^{k+1}d^{O(k^3)}$, where O denotes the big-O notation [Sip06, p. 252]. For more details on the algorithm, see chapter 15 in [BPR06].

A finite subdivision based on the transversal intersection of slices is defined in the following.

Definition 23 (Finite Subdivision)**.** Let $\mathcal{S} = \{\mathcal{S}^i |\ i \in \boldsymbol{k}\}$ be a collection of slice-families. We define a finite subdivision $E(\mathcal{S})$ by

$$e \in E(\mathcal{S}) \tag{3.18}$$

if and only if e is a cell of $\mathcal{S}$.

Occasionally, we also use finite subdivisions of extended cells $E_{\text{ex}}(\mathcal{S})$ defined by

$$e \in E_{\text{ex}}(\mathcal{S}) \tag{3.19}$$

if and only if e is an extended cell of $\mathcal{S}$.

We propose to use only functions that are decreasing along trajectories of the dynamical system Γ as subdivisioning functions, similar to Lyapunov functions, to obtain robustness of the subdivision. Indeed, the robustness is ensured as the vector field is transversal to the boundaries of the cells. This implies that there exists an arbitrary small perturbation of the vector field, such that it is still transversal to the boundary of the cells. The following definition is inspired by [Mey68].

Definition 24 (Decreasing Subdivisioning Function)**.** Let X be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous, and recall that $\text{Cr}(f)$ denotes the set of critical points of f. Then a real non-degenerate differentiable function $\varphi : X \to \mathbb{R}$ is said to be a subdivisioning function for f if

$$p \text{ is a critical point of } f \Leftrightarrow p \text{ is a critical point of } \varphi$$

$$L_f\varphi(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f^j(x) \tag{3.20a}$$

$$L_f\varphi(x) < 0 \qquad \forall x \in X \backslash \text{Cr}(f) \tag{3.20b}$$

and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \text{Cr}(f)$, where

$$||L_f\varphi(x)|| \geq \alpha ||x - p||^2. \tag{3.21}$$

We only require the vector field to be transversal to the level curves of a function φ, i.e., $L_f\varphi(x) = \langle \nabla\varphi(x), f(x) \rangle < 0$ for all $x \in X \backslash \text{Cr}(f)$ and require nothing about the sign of φ.

Note that a subdivisioning function from Definition 24 does not exist for systems with closed orbits. If a system has a closed orbit, it is necessary to subdivide the state space using forms rather than functions. This can be accomplished as shown in the following, where $0 \sim 2\pi$ means that 0 is identified with 2π, i.e., it is the same point.

$$\varphi_1 : [0, 2\pi]/_{0 \sim 2\pi} \to (0, 2\pi) \to (0, 2\pi) \subset \mathbb{R} : \theta \to \theta \tag{3.22a}$$

$$\varphi_2 : [-\pi, \pi]/_{(-\pi) \sim \pi} \to (-\pi, \pi) \to (0, 2\pi) \subset \mathbb{R} : \theta \to \theta + \pi \tag{3.22b}$$

Together φ_1 and φ_2 cover the entire sphere S^1, but one point is removed from both φ_1 and φ_2. This allows the abstraction of systems with closed orbits. The concept is illustrated in Figure 3.5.

In this thesis, we have only considered functions; hence, we cannot abstract systems with closed orbits.

In the next section, we present a procedure for generating a timed automaton from a subdivision, and show how the subdivisioning functions should be chosen to generate sound and complete abstractions.

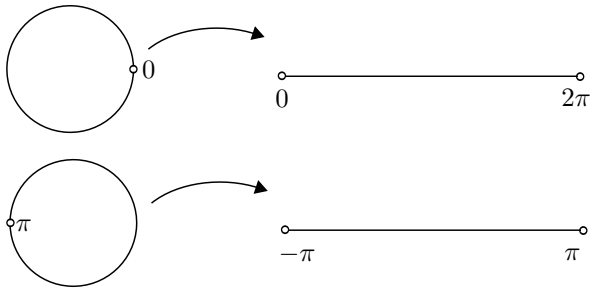


Figure 3.5: Illustration of φ_1 and φ_2.

Generation of Timed Automaton from Finite Subdivision

To generate an abstraction, we use an abstraction procedure similar to the procedure presented in [MB08]; nevertheless, we exclude the clock and constraints related to the time of traversing a cell. However, these can be added to improve accuracy. The reason why these clocks are removed is that they destroy the compositional structure of the timed automaton and requires computation of more guards and invariants.

A timed automaton $\mathcal{A}$ is generated from a finite subdivision $E(\mathcal{S})$ as follows.

Definition 25 (Generation of Timed Automaton)**.** Given a finite collection of slice-families $\mathcal{S} = \{\mathcal{S}^i |\ i \in \boldsymbol{k}\}$, and $\mathcal{T} = \{(\underline{t}^i_{g_i}, \overline{t}^i_{g_i}) \in \mathbb{R}^2_{\geq 0} |\ i \in \boldsymbol{k},\ g_i \in \{1, \ldots, |\mathcal{S}^i|\}\}$. The timed automaton $\mathcal{A}(\mathcal{S}, \mathcal{T}) = (E, E_0, C, \Sigma, I, \Delta)$ is defined by

- **Locations:** The locations of $\mathcal{A}$ are given by

$$E = E(\mathcal{S}). \tag{3.23}$$

  This means that a location $e_{(g,h)}$ is identified with the cell $e_{(g,h)} = \alpha^{-1}_{E(\mathcal{S})}(\{e_{(g,h)}\})$ of the subdivision $E(\mathcal{S})$, see Definition 14.

- **Clocks:** The set of clocks is $C = \{c^i |\ i \in \boldsymbol{k}\}$.

- **Alphabet:** The alphabet is $\Sigma = \{\sigma^i |\ i \in \boldsymbol{k}\}$.

- **Invariants:** In each location $e_{(g,h)}$, we impose an invariant

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \overline{t}^i_{g_i}. \tag{3.24}$$

- **Transition relations:** If a pair of locations $e_{(g,h)}$ and $e_{(g',h')}$ satisfy the following two conditions

  1. $e_{(g,h)}$ and $e_{(g',h')}$ are adjacent; that is $e_{(g,h)} \cap e_{(g',h')} \neq \emptyset$, and
  2. $g'_i \leq g_i$ for all $i \in \boldsymbol{k}$.

Then there is a transition relation

$$\delta_{(g,h)\to(g',h')} = (e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma^i, R_{(g,h)\to(g',h')}, e_{(g',h')}),$$

where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}^i_{g_i} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise.} \end{cases} \tag{3.25a}$$

Note that $g_i - g'_i = 1$ whenever a transition labeled σ^i is taken.

Let $i \in \boldsymbol{k}$. We define $R_{(g,h)\to(g',h')}$ by

$$c^i \in R_{(g,h)\to(g',h')} \tag{3.25b}$$

iff $g_i - g'_i = 1$.

From the definition, it is seen that if the i^{th} face is the common facet of the cells $e_{(g,h)}$ and $e_{(g',h')}$, then a transition is possible if the valuation of clock c^i is greater than $\underline{t}^i_{g_i}$. If the particular cells are not adjacent then no guard conditions are imposed.

If the set $\mathcal{S}$ is a singleton, i.e., $\mathcal{S} = \{\mathcal{S}_1\}$ then by slightly abusing the notation, we write $\mathcal{A}\left(\mathcal{S}_1, (\underline{t}, \bar{t})\right)$ instead of $\mathcal{A}\left(\{\mathcal{S}_1\}, \{(\underline{t}, \bar{t})\}\right)$. A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S}, \mathcal{T})$ has locations given by

$$E = E_{\text{ex}}(\mathcal{S}), \tag{3.26}$$

where a location $e_{\text{ex},g} \in E_{\text{ex}}(\mathcal{S})$ is associated with the extended cell $e_{\text{ex},g}$ generated by the slice-family $\mathcal{S}$; hence, $e_{\text{ex},g} = \alpha^{-1}_{E_{\text{ex}}(\mathcal{S})}(\{e_{\text{ex},g}\})$.

3.3 Properties of the Abstraction

The purpose of this section is to present selected properties of the abstraction. First, it is shown that the abstraction framework is compositional; second, it is shown when abstractions based on cells and extended cells are bisimilar. Then, conditions for generating sound and complete abstractions are provided. This is the main contribution and the most important property for determining the quality of the abstraction. Finally, results on refinement are presented.

The purpose of the abstraction is to verify a logic statement over predicates defined by regions of the state space; however, as the locations of the abstraction correspond to cells of the subdivision of the state space, level sets must be chosen such that the regions defined by predicates in temporal logic are subsets of a union of cells of the subdivision. Finally, the actual verification is conducted on the over-approximation of the regions involved in the temporal logic statement. This is illustrated in Figure 3.6, where the cell e_1 over-approximates the set X_0, and the verification is conducted as if X_0 was e_1.

Compositionality of the Abstraction

Under certain conditions it is possible to generate the timed automaton as a parallel composition of multiple timed automata [BK08, p. 48].

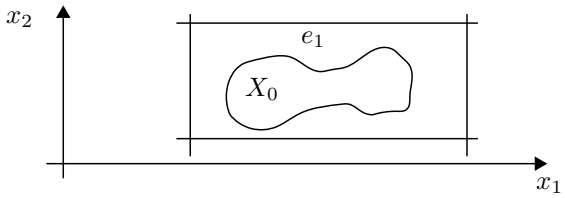


Figure 3.6: A set X_0 over-approximated by a cell e_1. The verification is accomplished using e_1, not X_0.

Definition 26 (Parallel Composition of Timed Automata)**.** The parallel composition of two timed automata, $\mathcal{A}_i = (E_i, E_{0,i}, C_i, \Sigma_i, I_i, \Delta_i)$ for $i = 1, 2$ with transition relations $(e_i, G_{i,e_i \to e_i'}, \sigma^i, R_{i,e_i \to e_i'}, e_i')$, is denoted by $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$ and is a timed automaton $(E, E_0, C, \Sigma, I, \Delta)$, where

- $E = E_1 \times E_2$;
- $E_0 = E_{0,1} \times E_{0,2}$;
- $C = C_1 \cup C_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $I : E \to \Psi(C)$, where $I(e_1, e_2) = I_1(e_1) \wedge I_2(e_2)$;
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations, where $(e, G_{e \to e'}, \sigma, R_{e \to e'}, e')$ is defined by the following
  1. If $\sigma \in \Sigma_1 \cap \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{1,e_1 \to e_1'} \wedge G_{2,e_2 \to e_2'}$, $R_{e \to e'} = R_{1,e_1 \to e_1'} \cup R_{2,e_2 \to e_2'}$, and $e' = (e_1', e_2')$.
  2. If $\sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{1,e_1 \to e_1'}$, $R_{e \to e'} = R_{1,e_1 \to e_1'}$, and $e' = (e_1', e_2)$.
  3. If $\sigma \notin \Sigma_1$ and $\sigma \in \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{2,e_2 \to e_2'}$, $R_{e \to e'} = R_{2,e_2 \to e_2'}$, and $e' = (e_1, e_2')$.

Proposition 3*: Let $\mathcal{A}_{\text{ex}}(\mathcal{S})$ be a timed automaton and let the slices of $\mathcal{S}$ be generated such that for each pair $(S^i_{g_i}, S^j_{g_j})$, with $i, j \in \boldsymbol{k}$, $g_i \in \{1, \ldots, |\mathcal{S}^i|\}$, $g_j \in \{1, \ldots, |\mathcal{S}^j|\}$, we have*

$$S^i_{g_i} \pitchfork S^j_{g_j} \neq \emptyset \qquad \forall i \neq j. \tag{3.27}$$

Then, $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is isomorphic to the parallel composition of k timed automata each generated by one slice-family $\mathcal{S}^i$ having an alphabet $\Sigma_i = \{\sigma^i\}$.

Remark 7*: A parallel composition of timed automata $\mathcal{A}_i(\mathcal{S}^i)$ for $i \in \boldsymbol{k}$ is similar to the intersection of slices in the slice-families $\mathcal{S}^i$. Therefore, the intersection of slices should be nonempty to let the locations of the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ be such a parallel composition, as stated in Proposition 3.*

The property that $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is isomorphic to the parallel composition of k timed automata is very important for computations, since it allows parallel verification of the

k timed automata each with only one clock. Furthermore, it makes it possible to sequentially add slice-families to the abstraction, to replace, and to refine slice-families to improve the accuracy of the abstraction.

The parallel composition of timed automata allows the sequential verification of the abstraction. We show this in terms of safety in the following.

Definition 27 (Safety)**.** Given a timed automaton $\mathcal{A}(\mathcal{S})$ and a set of unsafe locations E_{US}. The timed automaton $\mathcal{A}(\mathcal{S})$ is said to be safe if

$$\Phi_{\mathcal{A}(\mathcal{S})}([0,\infty), E_0) \cap E_{\text{US}} = \emptyset. \tag{3.28}$$

Proposition 4*: Let $\mathcal{A}_{\text{ex}}(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1)||\ldots||\mathcal{A}_k(\mathcal{S}^k)$ be a timed automaton and let the timed automaton $\mathcal{A}_1(\mathcal{S}^1)||\ldots||\mathcal{A}_j(\mathcal{S}^j)$ be safe, for some $j \in \boldsymbol{k}$. Then, $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is also safe.*

The compositional properties of the abstraction are exploited to verify a high dimensional system in Paper A on page 108.

Cells and Extended Cells

Under certain conditions, the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is bisimilar to the timed automaton $\mathcal{A}(\mathcal{S})$, bisimilarity is defined in Definition 75 on page 110. In the next proposition we say when the timed automata $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are related by bisimulation.

Proposition 5*: Let $\mathcal{S} = \{\mathcal{S}^i |\ i \in \boldsymbol{k}\}$ be a collection of slice-families, and φ^i be a subdivisioning function for $\mathcal{S}^i$. A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by extended cells is bisimilar to a timed automaton $\mathcal{A}(\mathcal{S})$ generated by cells if for each cell $e_{(g,h)}$ and each $i \in \boldsymbol{k}$*

$$e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i-1}) \neq \emptyset \qquad \forall\, h \text{ or} \tag{3.29a}$$

$$e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i-1}) = \emptyset \qquad \forall\, h. \tag{3.29b}$$

If (3.29) holds, then all cells in each extended cell have the same symbols on their outgoing transitions; hence, $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are bisimilar. The following example clarifies this proposition.

Example 5. To illustrate the use of Proposition 5 three different subdivisions of a two-dimensional state space are shown in Figure 3.7.

In the subdivisions shown in the left and right side of Figure 3.7, the conditions shown in (3.29) are satisfied for all g and h. In the subdivision shown in the middle side of Figure 3.7, the constraints shown in (3.29) are not satisfied for e.g. $g = (2,2,2)$ (shaded region), as $(\varphi^3)^{-1}(a^3_1)$ (inner black line) does not intersect all cells in $e_{\text{ex},(2,2,2)}$.

Soundness and Completeness

To ensure that the properties of an abstraction is not only valid for a particular choice of level sets, we impose the condition for any choice of regular values in the subdivision. To generate a timed automaton, it is required to devise a subdivision of the state space, and find a set of invariant and guard conditions. Therefore, we provide conditions under

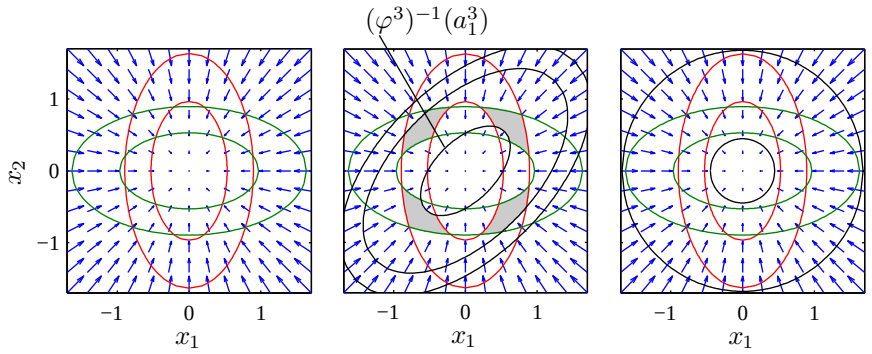


Figure 3.7: Illustration of three different subdivisions of a two-dimensional state space. The left and right subdivisions satisfy Proposition 5. The middle subdivision does not satisfy Proposition 5, as, e.g., the extended cell shaded with gray consists of cells, which have a different number of neighboring cells.

which an abstraction is sound or complete. Recall the definition of a sound and complete abstraction in Section 3.2.

It is chosen to present a sufficient condition for soundness, based on the decay rate (γs in Proposition 6) of a system, as numerous methods exist for calculating decay rates. The proposition is not found in the attached papers, but a similar condition is given in [SW11] for linear systems.

Proposition 6 (Sufficient Condition for Soundness)*: A timed automaton $\mathcal{A}(\mathcal{S},\mathcal{T})$ is a sound abstraction of the system $\Gamma = (X, f)$, if its invariants and guards satisfy*

$$\underline{t}_{S^i_{g_i}} \leq -\log\left(\frac{a^i_{g_i-1}}{a^i_{g_i}}\right)\frac{1}{\overline{\gamma}^i} \tag{3.30a}$$

$$\overline{t}_{S^i_{g_i}} \geq -\log\left(\frac{a^i_{g_i-1}}{a^i_{g_i}}\right)\frac{1}{\underline{\gamma}^i}, \tag{3.30b}$$

where

$$\overline{\gamma}^i = \arg\inf_{\gamma} |L_f\varphi^i(x)| \leq \gamma|\varphi^i(x)| \qquad \forall x \in S^i_{g_i} \tag{3.31}$$

$$\underline{\gamma}^i = \arg\sup_{\gamma} |L_f\varphi^i(x)| \geq \gamma|\varphi^i(x)| \qquad \forall x \in S^i_{g_i}. \tag{3.32}$$

Proposition 6 states that an abstraction is sound if all guards (given by $\underline{t}_{S^i_{g_i}}$) enable a transition out of a cell before any solution trajectory of the dynamical system can traverse the cell, and if all invariants (given by $\overline{t}_{S^i_{g_i}}$) enable a run to stay in a cell longer than any solution trajectory of the dynamical system uses to traverse the cell.

The values of guards of $\underline{t}_{S^i_{g_i}}$ and $\overline{t}_{S^i_{g_i}}$ can be algorithmically generated. This is shown for linear systems in Corollary 3 on page 37.

Proposition 7*: Given a dynamical system $\Gamma = (X, f)$, a collection of subdivisioning functions $\{\varphi^i |\ i \in \boldsymbol{k}\}$, a collection of regular values $\{a^i_{g_i} |\ i \in \boldsymbol{k},\ g_i \in \{1, \ldots, |\mathcal{S}^i|\}\}$ generating $\mathcal{S}$, and*

$\mathcal{T} = \{(\underline{t}^i_{g_i}, \bar{t}^i_{g_i})|\ i \in \boldsymbol{k},\ g_i \in \{1, \dots, |\mathcal{S}^i|\}\}$. The timed automaton $\mathcal{A}(\mathcal{S}, \mathcal{T})$ is a complete abstraction of Γ if and only if for any $i \in \boldsymbol{k}$

1. *for any $g \in \mathcal{G}(\mathcal{S})$ (see the definition of $\mathcal{G}(\mathcal{S})$ in Definition 21), such that $g_i \geq 2$ there exists a time $t^i_{g_i}$ such that for all $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$*

$$\phi_\Gamma(t^i_{g_i}, x_0) \in (\varphi^i)^{-1}(a^i_{g_i - 1}) \tag{3.33}$$

   and

2. *$\bar{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$.*

Proof. See Appendix A of Paper B. □

Proposition 7 states that it takes the same time for all solutions to propagate between two level sets in of a complete abstraction. This time must be used for both invariant and guard conditions.

Proposition 7 does not provide a straightforward method for computing a complete abstraction, as the condition is not numerically tractable. Therefore, we rephrase (3.33) as a relation between the level sets of the subdivisioning function and its Lie derivative. This is given in the following theorem.

Theorem 1: Let $\Gamma = (X, f)$ be a dynamical system. There exists a complete abstraction of Γ if and only if there exists a collection of subdivisioning functions $\{\varphi^i|\ i \in \boldsymbol{k}\}$, such that for any φ and any regular value $a \in \mathbb{R}$ there exists $b \in \mathbb{R}$ such that

$$\{x \in \mathbb{R}^n|\ \varphi(x) - a = 0\} \subseteq \{x \in \mathbb{R}^n|\ L_f\varphi(x) - b = 0\}. \tag{3.34}$$

Proof. See Paper B on page 135. □

Corollary 1: From (7.26) in the proof of Theorem 1, the time $\bar{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$ in Proposition 7 can be calculated for a slice $S^i_{g_i} = (\varphi^i)^{-1}([a,\ b])$ by simulating a trajectory from any point $x_0 \in (\varphi^i)^{-1}(b)$ and determining the time $t^i_{g_i}$, when $\phi_\Gamma(t^i_{g_i}, x_0) \in (\varphi^i)^{-1}(a)$.

We identify a necessary and sufficient relation between two polynomials $\varphi, \psi \in \mathbb{R}[\underline{x}]$ satisfying (3.34).

In this work, we are only interested in using irreducible polynomials to subdivide the state space, as they have a closed and connected variety [Has07, p. 90]. By using irreducible polynomials, the generated slices are connected components, as desired.

Definition 28 (Irreducible Polynomial [Has07])**.** A polynomial $p \in \mathbb{R}[\underline{x}]$ is called irreducible if it is nonconstant and there exist no two nonconstant polynomials $p_1, p_2 \in \mathbb{R}[\underline{x}]$ such that

$$p = p_1 p_2. \tag{3.35}$$

Theorem 2: Let $\varphi, \psi \in \mathbb{R}[\underline{x}]$. For any regular value $a \in \mathbb{R}$, there exists $b \in \mathbb{R}$ such that

$$\{x \in \mathbb{R}^n|\ \varphi(x) - a = 0\} \subseteq \{x \in \mathbb{R}^n|\ \psi(x) - b = 0\}, \tag{3.36}$$

where $\varphi - a$ is irreducible if and only if

$$\psi = c_0 + c_1\varphi + c_2\varphi^2 + c_3\varphi^3 + \cdots. \tag{3.37}$$

Proof. See Paper B, page 137. □

Essentially, (3.37) states that ψ must be a polynomial in φ to ensure that any level set of φ is a subset of some level set of ψ.

Based on Theorem 2, we rephrase Theorem 1 as follows.

Theorem 3: Let $\Gamma = (X, f)$ be a dynamical system. There exists a complete abstraction of Γ if and only if there exists a collection of subdivisioning functions $\{\varphi^i | \, i \in \boldsymbol{k}\}$, such that

$$L_f \varphi^i = c_0 + c_1 \varphi^i + c_2 (\varphi^i)^2 + c_3 (\varphi^i)^3 + \cdots . \tag{3.38}$$

This is a very important result that characterizes *optimal* subdivisioning functions.

Theorem 3 is utilized to generate both sound and complete abstractions, by generating partitioning functions that are as close as possible to satisfying (3.38). This gives the best abstraction, although it is not necessarily complete.

Refinement of Abstraction

To ensure that an abstraction can approximate the reachable set of Γ with any desired accuracy, it should be refinable according to the following definitions.

Definition 29 (Refinement of subdivision)**.** Let the subdivision $E(\mathcal{S})$ be generated by the slice-families $\mathcal{S} = \{\mathcal{S}^i | \, i \in \boldsymbol{k}\}$. Then the subdivision $E(\tilde{\mathcal{S}})$ is a refinement of $E(\mathcal{S})$ if and only if for any $e \in E(\mathcal{S})$ there is a cell $\tilde{e} \in E(\tilde{\mathcal{S}})$ such that

$$\tilde{e} \subseteq e. \tag{3.39}$$

Definition 30 (Refinable Abstraction)**.** An abstraction $\mathcal{A}(\mathcal{S})$ of a system Γ is said to be refinable if for all $\epsilon > 0$ there exists an abstraction $\mathcal{A}(\tilde{\mathcal{S}})$, where $E(\tilde{\mathcal{S}})$ is a refinement of $E(\mathcal{S})$, such that for all $\tilde{e} \in E(\tilde{\mathcal{S}})$

$$\tilde{e} \subseteq B(\epsilon), \tag{3.40}$$

where $B(\epsilon)$ is a ball with radius ϵ.

Note that the ball $B(\epsilon)$ need not be centered at 0. The least ϵ that satisfies (3.40) for all $\tilde{e} \in E(\tilde{\mathcal{S}})$ is called the radius of the subdivision. We see that combining the notion of complete abstraction and refinable abstraction yields the following corollary.

Corollary 2: If the system Γ is abstracted by a complete and refinable abstraction $\mathcal{A}(\mathcal{S})$, then for all $\epsilon > 0$ there exists a subdivision $E(\tilde{\mathcal{S}})$ such that for all $t \in [t_1, t_2]$

$$\alpha_K^{-1} \left(\Phi_{\mathcal{A}(\tilde{\mathcal{S}})}(t, E_0) \right) \subseteq \phi_\Gamma(t, X_0) + B(\epsilon). \tag{3.41}$$

The smallest ϵ such that (3.41) is satisfied is called the accuracy of the abstraction. Corollary 2 states that a complete and refinable abstraction can approximate the reachable states of a system Γ arbitrarily close. In conclusion, to get any desired radius of the subdivision, all cells of its subdivision $E(\mathcal{S})$ should converge towards points. In the next proposition, we answer the question of minimal number of slice-families necessary to construct a refinable abstraction.

Proposition 8 (Necessary Condition for Refinable Abstraction)*: If $\mathcal{A}(\mathcal{S})$ is a refinable abstraction of a system Γ, then $\mathcal{S}$ is a collection of n slice-families.*

Proof. See Paper A on page B. □

Note that this is necessary to be able to shrink a cell to a point.

3.4 Algorithmic Generation of Abstraction

Algorithms are developed for abstracting linear systems using linear matrix inequalities (LMIs), and sum of squares (SOS) conditions for finding the best subdivisioning functions for polynomial systems. Most of these conditions are alike. Therefore, we only show LMIs for generating invariant conditions for linear systems.

Invariant conditions can be generated from Corollary 3. Note that a more general condition can be set up, but this condition is presented to clarify presentation.

Corollary 3*: Suppose $\Gamma = (X, f)$ is a stable linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$, and $P^i \succ 0$. Let $L_f\varphi^i = -x^{\mathrm{T}} Q^i x$, and define $\underline{\gamma}^i$ as the solution to the following optimization problem*

$$\text{maximize } \underline{\gamma}^i \tag{3.42a}$$

$$\text{subject to } \underline{\gamma}^i P^i - Q^i \preceq 0, \qquad \underline{\gamma}^i > 0. \tag{3.42b}$$

Then for any location $e_{(g,h)} \in E$, the invariant is

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \bar{t}_{S^i_{g_i}}, \text{ where} \tag{3.43a}$$

$$\bar{t}_{S^i_{g_i}} \geq -\log\left(\frac{a^i_{g_i - 1}}{a^i_{g_i}}\right) \frac{1}{\underline{\gamma}^i}. \tag{3.43b}$$

In Corollary 3, it is seen that the same decay rate $\underline{\gamma}^i$ is used for all invariants involving clock c^i. This makes the computations of the abstraction especially simple. A geometric interpretation of the optimization problem is shown in Figure 3.8. The minimum value of $L_f\varphi$ can be determined by finding the smallest value a, where $(L_f\varphi)^{-1}(a)$ intersects a boundary of a slice.

To generate complete abstractions, we cannot search for a polynomial φ satisfying the relation shown in (3.38) using linear or sum of squares optimization problems, as the coefficients of the polynomials would appear in different powers. Therefore, we assume that $L_f\varphi$ is an affine function of φ.

Proposition 9*: Given a dynamical system $\Gamma = (X, f)$, $\underline{c}_0, \underline{c}_1, \bar{c}_0, \bar{c}_1 \in \mathbb{R}$, and a subdivisioning function φ such that*

$$\underline{c}_0 + \underline{c}_1\varphi(x) \leq L_f\varphi(x) \leq \bar{c}_0 + \bar{c}_1\varphi(x) \qquad \forall x. \tag{3.44}$$

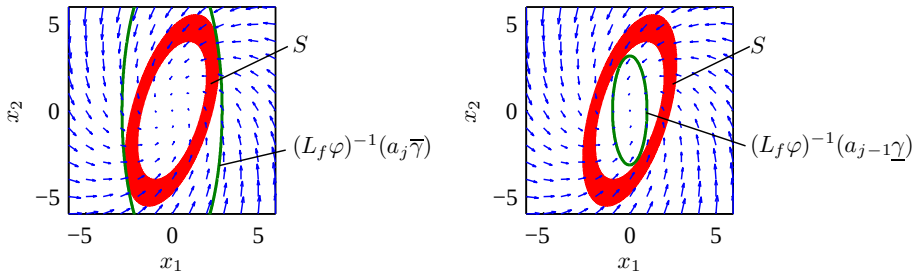


Figure 3.8: Illustration of a slice $S = \varphi^{-1}([a_{j-1}, a_j])$ (red) and the level sets $(L_f\varphi)^{-1}(a_{j-1}\underline{\gamma})$ and $(L_f\varphi)^{-1}(a_j\overline{\gamma})$ (green and dashed) intersecting the slice S.

Let $\mathrm{CrV}(\varphi)$ *be the set of critical values of* φ*. Then for any pair of regular values* $a_1 < a_2$*, where* $\mathrm{CrV}(\varphi) \cap [a_1,\, a_2] = \emptyset$*, and any* $x_0 \in \varphi^{-1}(a_2)$ *there exists* $t > 0$ *that satisfies*

$$\textit{sign}(\underline{c}_1 a_2 + \underline{c}_0)t \leq \mathrm{sign}(\underline{c}_1 a_2 + \underline{c}_0)\frac{1}{\underline{c}_1}\log\left(\frac{\underline{c}_0}{\underline{c}_1}\right)\log\left(\frac{a_2}{a_1}\right), \textit{ and} \tag{3.45a}$$

$$\textit{sign}(\overline{c}_1 a_2 + \overline{c}_0)t \geq \mathrm{sign}(\overline{c}_1 a_2 + \overline{c}_0)\frac{1}{\overline{c}_1}\log\left(\frac{\overline{c}_0}{\overline{c}_1}\right)\log\left(\frac{a_2}{a_1}\right) \tag{3.45b}$$

such that

$$\phi_\Gamma(t, x_0) \in \varphi^{-1}(a_1). \tag{3.46}$$

Proof. See Paper B on page 139. □

Note that $L_f\varphi(x)$ is required to be negative for any $x \in X \backslash \mathrm{Cr}(f)$. Therefore, $(\overline{c}_1 a_1 + \overline{c}_0)$ and $(\underline{c}_1 a_1 + \underline{c}_0)$ should also be negative; hence, (3.45a) gives a lower bound on t and (3.45b) gives an upper bound on t. Furthermore, it is seen from (3.45) that for $\underline{c}_0 = \overline{c}_0$ and $\underline{c}_1 = \overline{c}_1$, the abstraction generated by φ is complete. Otherwise, we minimize the time interval given by (3.45), to obtain the most accurate sound abstraction. This is accomplished by minimizing the upper bound and maximizing the lower bound on $L_f\varphi$ given in (3.44). The procedure for this is provided next.

First, we make the polynomials $\varphi, L_f\varphi$ homogeneous, to eliminate the constants $\underline{c}_0$ and $\overline{c}_0$.

Definition 31 ([Mar08])**.** A homogeneous polynomial (or form) is a polynomial where all monomials have the same total degree d.

We can transform a polynomial into a homogeneous polynomial as follows.

Lemma 1 ([Mar08])**.** *Let* f *be any polynomial in* $\mathbb{R}[\underline{x}]$ *of degree less than or equal to* d*. Then*

$$\bar{f}(x_0, \ldots, x_n) = x_0^d f(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}) \tag{3.47}$$

is a homogeneous polynomial of degree d*.*

It is very important to choose the degree d correctly, when generating the homogeneous polynomial, to ensure the following property.

Lemma 2. *Let f and $\bar{f}$ be related by* (3.47)*. If d is even, then $f(x) \geq 0$ for all $x \in \mathbb{R}^n$ if and only if $\bar{f}(x) \geq 0$ for all $x \in \mathbb{R}^{n+1}$.*

Proposition 10*: Let $\varphi, L_f\varphi \in \mathbb{R}[\underline{x}]$ and $c_0, c_1 \in \mathbb{R}$. Then*

$$L_f\varphi(x) \leq c_0 + c_1\varphi(x) \qquad \forall x \in \mathbb{R}^n \tag{3.48}$$

if and only if

$$\overline{L_f\varphi}(y) \leq 1 + c_1\bar{\varphi}(y) \qquad \forall y \in \mathbb{R}^{n+1}, \tag{3.49}$$

where $\bar{\varphi}, \overline{L_f\varphi} \in \mathbb{R}[x_0, \ldots, x_n]$ are homogeneous polynomials of $\varphi, L_f\varphi$.

Proof. See Paper B on page 140. □

From Proposition 10, we see that by considering homogeneous polynomials, only one decision variable is needed to obtain each bound of (3.44).

We use sum of squares (SOS) optimization problems to compute polynomial subdivisioning functions for polynomial systems. The following explanation of sum of squares polynomials is based on [Par03, BCR98].

Definition 32. A polynomial $p \in \mathbb{R}[\underline{x}]$ is called sum of squares if

$$p = \sum_{i=1}^{k} p_i^2 \tag{3.50}$$

for some polynomials $p_i \in \mathbb{R}[\underline{x}]$ with $i = 1, \ldots, k$.

The set of sum of squares polynomials is a subset of nonnegative polynomials [Par03], which can be treated using semidefinite programming, as described below. We denote the set of sum of squares polynomials in n variables by Σ_n.

The existence of a sum of squares decomposition of a polynomial $p \in \mathbb{R}[\underline{x}]$ of degree d can be expressed as a semidefinite programming feasibility problem. Therefore, the formulation of a problem as sum of squares makes the problem computationally tractable; however, the number of decision variables in the program is

$$N = \binom{n + 2d}{2d}. \tag{3.51}$$

In the search of sum of squares polynomials, it is exploited that the existence of an SOS decomposition of a polynomial p is equivalent to the existence of a matrix $Q = Q^{\mathrm{T}} \succeq 0$ such that

$$p = Z^{\mathrm{T}} Q Z, \tag{3.52}$$

where Z is a vector of monomials of degree less than or equal to half the degree of p.

Let $k, l \in \mathbb{Z}_{>0}$, let $\alpha_{i,j} \in \mathbb{R}[\underline{x}]$ for $(i,j) \in k \times l$, and $w_j \in \mathbb{R}$. An SOS programming problem is

$$\begin{aligned} &\underset{(c_1,\ldots,c_l)\in\mathbb{R}^l}{\text{minimize}} \sum_{j=1}^{l} w_j c_j \text{ subject to} \\ &\alpha_{i,0} + \sum_{j=1}^{l} \alpha_{i,j} c_j \in \Sigma_n \forall i = 1, \ldots, k. \end{aligned} \tag{3.53}$$

It is seen that an SOS programming problem is a minimization of a linear cost subject to SOS feasibility constraints.

The main issue with sum of squares polynomials is that a polynomial may be nonnegative, even though it cannot be represented as a sum of squares [BCR98].

We find the subdivisioning function that gives the best abstraction, via a sum of squares optimization problem. Note that this is very similar to the problem of finding the maximum decay rate of a system, which can be formulated as a generalized eigenvalue problem. This problem can be solved using the bisection method.

In the considered problem, we do not assume the system to be stable nor unstable; hence, we cannot assume any sign of the decay rate. However, to avoid complicating the optimization problems, we assume that the decay rate is positive (in the optimization problem $\gamma > 0$), but if $\gamma < 0$ all maximizations should just be replaced with minimizations and vice versa.

Proposition 11*: Suppose $\Gamma = (X, f)$ is a polynomial system, then the subdivisioning function φ, minimizing the upper bound of* (3.49) *is given by the following optimization problem*

$$\begin{aligned} &\textit{max } \gamma \textit{ subject to} \\ 1 + \gamma\varphi - L_f\varphi &\in \Sigma_n \\ -L_f\varphi &\in \Sigma_n \\ \gamma &> 0 \end{aligned} \tag{3.54}$$

and the subdivisioning function φ, maximizing the lower bound of (3.49) *is given by the following optimization problem*

$$\begin{aligned} &\textit{min } \gamma \textit{ subject to} \\ -1 - \gamma\varphi + L_f\varphi &\in \Sigma_n \\ -L_f\varphi &\in \Sigma_n \\ \gamma &> 0. \end{aligned} \tag{3.55}$$

These optimization problems can be solved using the bisection method [BG93]. Note that the previous optimization problems can be solved in tools such as SOSTOOLS [PPSP05]. This is a tool that transforms sum of squares optimization problems into semidefinite programs (SDPs), which can be directly solved using, e.g., SeDuMi. Solving SDPs is polynomial time [GM12]; hence, the sum of squares optimization problem can be solved in polynomial time, with the number of decision variables given by (3.51).

In Paper B on page 142 we also pose the optimization problem using LMIs, which can be used for linear systems with quadratic subdivisioning functions.

This finalizes the description of the abstraction method. In the next section, the method is adapted to abstract mechanical systems.

3.5 Abstractions for Mechanical Systems

In this section an abstraction method is presented for abstracting mechanical systems. To allow abstraction of mechanical systems, the requirements for the subdivisioning functions are relaxed, and reduction techniques for mechanical systems are used to generate the subdivisioning functions.

Definition 33 (Transversal Subdivision)**.** Let X be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous and let $\mathrm{Cr}(f)$ be the set of critical points of f (equilibria). Let $\Phi = \{\varphi_i : X \to \mathbb{R} |\ i \in \boldsymbol{k}\}$ be a set of real differentiable functions, and let $\mathbb{A} = \{A_i |\ i \in \boldsymbol{k}\}$ be a collection of sets of regular values. The finite subdivision $E(\Phi, \mathbb{A})$ is said to be transversal (we call it a transversal subdivision) if for each cell $e \in E(\Phi, \mathbb{A})$ there is a subdivisioning function $\varphi_i \in \Phi$ such that

$$L_f \varphi_i(x) \neq 0 \qquad \forall x \in e \backslash \mathrm{Cr}(f) \tag{3.56a}$$

and for all $i \in \boldsymbol{k}$

$$L_f \varphi_i(x) = 0 \qquad \forall x \in \mathrm{Cr}(f). \tag{3.56b}$$

It is seen from (3.56a) that at least one subdivisioning function has to have nonzero gradient in each cell (3.56a); hence, the vector field should be transversal to the level sets of at least one subdivisioning function. This is important in the generation of time information for the abstraction.

The abstraction is accomplished in the following steps

(A) Discard all dissipation of the system and subdivision the state space of the conservative system using tangential and transversal manifolds.

(B) Add dissipation and select level sets to obtain a transversal subdivision.

(C) Generate a timed automaton abstracting the system.

We show in Proposition 13 that a transversal subdivision generated by n subdivisioning functions can be realized for integrable systems using the presented procedure.

First, we consider a conservative mechanical system, as this enables the identification of cyclic coordinates and first integrals (constants of motion). The constants of motion are functions with level sets being tangential manifolds; hence, they are used as subdivisioning functions. Following this identification, the model is reduced using Routh reduction [Gol60], and the reduced space is subdivided using transversal manifolds, which are generated via action-angle coordinates. Finally, we add dissipation to the system. This implies that the system trajectories no longer are confined to a certain constant of motion. Instead, the system trajectories traverse the manifolds according to dynamics described by the dissipation. This subdivision is shown to be transversal.

Discretizing Conservative Mechanical System

The aim of this subsection is to provide guidance for finding $2n$ mutually transversal subdivisioning functions for a conservative mechanical system with n degrees of freedom. It is required to find $2n$ mutually transversal subdivisioning functions, i.e., functions whose gradients are linearly independent at each point (except of critical points), to obtain arbitrary accuracy of the abstraction. The method consists of the following steps

1. Identify cyclic coordinates from the Lagrangian.
2. Find tangential manifolds via Noether's theorem.
3. Reduce the system using Routh reduction.
4. Find transversal manifolds for the reduced system using action-angle coordinates.

We assume that the mechanical system with n degrees of freedom is described by n Euler-Lagrange equations of motion in generalized coordinates.

Identification of Cyclic Coordinates

Recall that a coordinate q^i is said to be cyclic if the Lagrangian of a system does not depend on it.

From the Lagrangian of a system, it is seen that $\partial L/\partial q^i = 0$ if q^i is cyclic; hence, the generalized momentum $\partial L/\partial \dot{q}^i$ is constant. This means that cyclic coordinates identify symmetries of the system, where a symmetry is a transformation that generates a displacement under which the system is invariant, e.g., a translation along a cyclic coordinate. Therefore, each cyclic coordinate should be subdivided independently of the other coordinates, i.e., if q^i is a cyclic coordinate then

$$\varphi : (q, \dot{q}) \mapsto q^i \tag{3.57}$$

should be used as subdivisioning function. The cyclic coordinate should be discarded in the remainder of the subdivisioning procedure.

Identification of Tangential Manifolds

We are interested in constructing the tangential manifolds without the use of solutions of the differential equations. This motivates the identification of tangential manifolds, via the Euler-Lagrange equations. The presented method may not identify $2n - 1$ constants of motion; however, the tangential manifolds are identified globally. The number of constants of motion that one can find for a given system is not a priori known.

Following [Arn89, p. 207], the function H is a first integral of the Hamiltonian phase flow with Hamiltonian function H. This implies that we can always find one constant of motion: the Hamiltonian. The Hamiltonian function should be used as a tangential subdivisioning function

$$\varphi(q, \dot{q}) = H(q, \dot{q}). \tag{3.58}$$

An integrable system has, per definition, n linear independent tangential manifolds. These are also called functionally independent constants of motion. The Poisson bracket

is used in the definition of an integrable system. Recall that given two smooth real-valued functions A and B defined on the phase space of a Hamiltonian system, the canonical Poisson bracket of A and B is defined by

$$\{A, B\} = \sum_{i=1}^{N} \left(\frac{\partial A}{\partial q^i} \frac{\partial B}{\partial p_i} - \frac{\partial B}{\partial q^i} \frac{\partial A}{\partial p_i} \right), \tag{3.59}$$

where (q^i, p_i) are conjugate pairs of canonical coordinates [Mar92].

Definition 34 (Integrable System)**.** A Hamiltonian systems in a $2n$-dimensional symplectic manifold is said to be integrable (in the Arnold-Liouville sense) if there exist n functionally independent constants of motion that are in involution, meaning that they pairwise satisfy

$$\{F_i, F_j\} = 0 \quad \forall i, j. \tag{3.60}$$

Constants of motion can be found, by using the symmetries of the system, given by the cyclic coordinates, according to the following theorem [Arn89, p. 88].

Theorem 4 (Noether's Theorem)*: Let M be a smooth manifold, $L : TM \to \mathbb{R}$ a smooth function on its tangent bundle TM. If the system (M, L) admits the one-parameter group of diffeomorphisms $h_s : M \to M$, $s \in \mathbb{R}$, then the lagrangian system of equations corresponding to L has a first integral $I : TM \to \mathbb{R}$. In local coordinates q on M the integral I is written in the form*

$$I(q, \dot{q}) = \frac{\partial L}{\partial \dot{q}} \frac{dh_s(q)}{ds}\Big|_{s=0}. \tag{3.61}$$

From the theorem, it is seen that we can find one constant of motion per cyclic coordinate, as the generalized momentum $\partial L / \partial \dot{q}^i$ is constant if q^i is cyclic; hence, the generalized momentum should be used as a tangential subdivisioning function

$$\varphi(q, \dot{q}) = \frac{\partial L}{\partial \dot{q}^i}. \tag{3.62}$$

In relation to Theorem 4, let $M = \mathbb{R}^n$ and let the first coordinate be a cyclic coordinate, then $h_s : (q^1, \ldots, q^n) \mapsto (q^1 + s, \ldots, q^n)$ is a one-parameter group. Note that symmetry under translation corresponds to momentum conservation, symmetry under rotation to angular momentum conservation, symmetry in time to energy conservation [But05].

Reduction of the System

The remaining subdivision should be conducted on a reduced state space, given by the following theorem, which can be used to restrict the dynamics of a system to a lower dimensional surface using constants of motion, [LCV10].

Theorem 5 (Routh Reduction)*: Let $L : \mathbb{R}^{2n} \to \mathbb{R}$ be the Lagrangian for a system with n degrees of freedom. Assume that q^1 is a cyclic coordinate and that locally $\partial^2 L / \partial \dot{q}^1 \partial \dot{q}^1 \neq 0$ so that $\dot{q}^1$ can be expressed as $\dot{q}^1 = \varrho(q^2, \ldots, q^n, \dot{q}^2, \ldots, \dot{q}^n)$. Consider the Routhian $R^\mu : \mathbb{R}^{2(n-1)} \to \mathbb{R}$ defined as the function $R^\mu = L - \dot{q}^1 \mu$, where all instances of $\dot{q}^1$ are substituted by the function*

ϱ and momentum $p_1 = \mu$. The Routhian is now interpreted as the Lagrangian for a system with $(n-1)$ degrees of freedom $(q^2, \ldots, q^n)$.

Any solution $(q^1(t), \ldots, q^n(t))$ of the Euler-Lagrange equations of motion

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}^i} - \frac{\partial L}{\partial q^i} = 0, \qquad i = 1, \ldots, n \tag{3.63}$$

with momentum $p_1 = \mu$, projects onto a solution $(q^2(t), \ldots, q^n(t))$ of the Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial R^\mu}{\partial \dot{q}^k} - \frac{\partial R^\mu}{\partial q^k} = 0, \qquad k = 2, \ldots, n. \tag{3.64}$$

Conversely, any solution of the Euler-Lagrange equations for R^μ can be lifted to a solution of the Euler-Lagrange equations for L with momentum $p_1 = \mu$.

Using Theorem 5, we can obtain Euler-Lagrange equations of reduced dimension, which should be used in the generation of the transversal manifolds.

The idea of Routh reduction is to use the constants of motion as coordinates in the system description. This enables the system to be analyzed using fewer coordinates, as the system has no dynamics in the coordinates given by the constants of motion. The concept is shown in Figure 3.9.

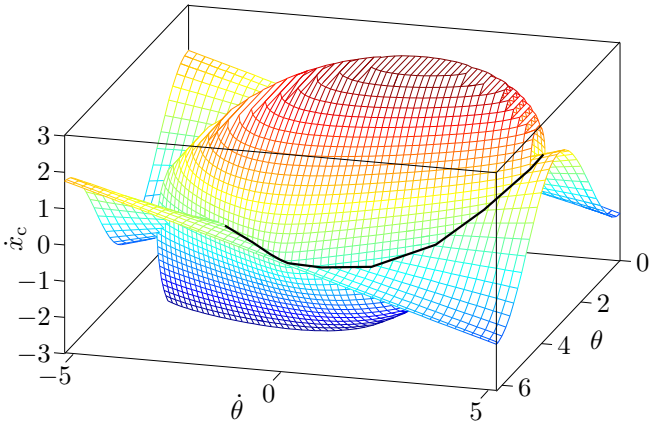


Figure 3.9: The two surfaces are level sets of the constants of motion. The black line is the simulated trajectory.

The figure illustrates two constants of motion and a solution trajectory (black line) that is located at their intersection; hence, the solution can be described using only one coordinate (apart from the constants of motion).

Identification of Transversal Manifolds

We have not found general method for finding transversal manifolds; however, for integrable systems, we can find transversal manifolds via the use of action-angle coordinates.

Theorem 6 ([JS98])*: Consider a completely integrable Hamiltonian system with constants of motion $C_1(q,p) = H(q,p), C_1(q,p), \ldots, C_n(q,p)$ which are in involution. The hypersurfaces given by sets of constants $c = \{c_i |\, i \in \boldsymbol{n}\}$*

$$S(c) = \{(q,p) \in T^*\mathbb{Q} |\, C_i(q,p) = c_i, i = 1, \ldots, n\} \tag{3.65}$$

are invariant under the flow of the Hamiltonian system. If $S(c)$ is compact and connected, then $S(c)$ can be mapped in a diffeomorphic way on a n-torus $T^n = S^1 \times \cdots \times S^1$. Each circle can be described by an angle coordinate $\theta_i(t)$ with dynamics

$$\frac{d\theta_i}{dt} = \Omega_i(c), \qquad i = 1, \ldots, n. \tag{3.66}$$

From the theorem we see that for integrable systems one can find a coordinate system, where n coordinates are given by constants of motion and n coordinates which are independent of each other and are given by trivial dynamics. For each action-angle θ_i, a transversal subdivision function

$$\varphi_i(q,p) = \theta_i \tag{3.67}$$

should be used in the subdivisioning of the state space. For details in the synthesis of the coordinate transformation, see [JS98].

Note that the proposed method does not provide $2n$ linear independent subdivisioning functions for all systems; however, for integrable systems they can be found via Theorem 6. Therefore, the proposed subdivision can be applied to partly subdivide a state space, and then the remaining part of the state space can be subdivided using, e.g., hypercubes as used in most other abstraction procedures.

Obtaining Transversal Subdivision

The final step of the subdivisioning procedure is to check if the subdivision is transversal. We show that a transversal subdivision can always be found for integrable systems.

Proposition 12*: Let the system (M, L) be defined as shown in Theorem 4 with first integral in local coordinates (U, ς)*

$$I(q,\dot{q}) = \frac{\partial L}{\partial \dot{q}} \frac{dh_s(q)}{ds}. \tag{3.68}$$

Then by adding external forces Q to the system, the time derivative of I becomes

$$\frac{d}{dt} I(q,\dot{q}) = Q \frac{dh_s(q)}{ds}. \tag{3.69}$$

Proof. See Paper C on page 161. □

Proposition 13*: Let (M, L) be an integrable system, and let $k = 2n$; thus, $\boldsymbol{k} = \{1, \ldots, 2n\}$. Then there exists a collection of nonempty sets of regular values $\mathbb{A} = \{A_i | i \in \boldsymbol{k}\}$ for the subdivisioning functions $\Phi = \{\varphi_i(q,\dot{q}) |\, i \in \boldsymbol{k}\}$ (see Theorem 6), such that the generated subdivision $E = (\Phi, \mathbb{A})$ is transversal.*

Proof. See Paper C on page 162 □

This implies that we can obtain a transversal subdivision for mechanical systems per construction.

3.6 Conclusion

An abstraction method for verifying TCTL specifications for Morse-Smale systems is presented in this chapter. It is shown that verification of TCTL specifications is possible using complete abstractions, and that universally quantified TCTL specifications can be verified by sound abstractions. We provide optimization problems for generating the subdivision of the state space used in the abstraction procedure. However, a complete solution to the generation of the subdivision is not provided.

Finally, we have provided an initial attempt to abstract mechanical systems using a similar abstraction method. This method is however not fully developed yet.

4 Safety Guarantees for Continuous Systems

The purpose of this chapter is to present a compositional method for verifying the safety of continuous systems, and a method for designing controllers that ensures safety of control systems. Both methods are based on the barrier certificate method.

The chapter is an extract of the results from Paper D-F. The compositional verification method is set up in Paper D and refined in Paper E. Paper F presents a new definition of control barrier function and provides an associated controller design method. This chapter gives answers to the following problems.

Problem 3*: Given a dynamical system decomposed into k subsystems, a set of initial states X_0, and a set of unsafe states X_u. Determine if there exists a trajectory initialized in X_0 that reaches X_u.*

Problem 4*: Given a control system, a set of initial states X_0, and a set of unsafe states X_u. Design a control law such that no trajectories initialized in X_0 can reach X_u.*

The chapter is organized as follows. Section 4.1 presents conditions for verifying safety using barrier certificates. The conditions are rephrased in a compositional manner in Section 4.2, algorithms for finding the barrier certificates are presented in Section 4.3, and Section 4.4 elaborates on the restrictions of the compositional method. Finally, a method for designing safe controllers is presented in Section 4.5, followed by conclusions in Section 4.6.

Notation

Let $k \in \mathbb{N}$. Given $x = (x_1, \ldots, x_k) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k}$, with $x_i \in \mathbb{R}^{n_i}$, we define $\hat{x}_i \equiv (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. Similarly, given a sequence of maps $(h_1, \ldots, h_k)$, we define $\hat{h}_i \equiv (h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_k)$.

4.1 Safety Verification using Barrier Certificates

This section presents the barrier certificate method that can be used for verifying the safety of a dynamical system.

We consider a continuous system given as a system of ordinary differential equations

$$\dot{x} = f(x, d), \tag{4.1}$$

where $x \in \mathbb{R}^n$ is the state and $d \in D \subseteq \mathbb{R}^r$ is the disturbance input.

For some measurable and essentially bounded disturbance function $\bar{d} : \mathbb{R}_{\geq 0} \to D$, i.e., $\bar{d} \in \mathcal{L}_\infty(\mathbb{R}_{\geq 0}, D)$, we denote the solution of the Cauchy problem (4.1) with initial state x_0 on an interval $[0, T]$ by $\phi^{\bar{d}}_{x_0}$, i.e.,

$$\frac{d\phi^{\bar{d}}_{x_0}(t)}{dt} = f\left(\phi^{\bar{d}}_{x_0}(t), \bar{d}(t)\right) \tag{4.2}$$

for almost all $t \in [0, T]$ and $\phi^{\bar{d}}_{x_0}(0) = x_0$. We denote the set of solutions from all initial conditions x_0 in X_0 by $\phi^{\bar{d}}_{X_0}$.

We consider a dynamical system given by $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$, where $f : \mathbb{R}^{n+r} \to \mathbb{R}^n$ is continuous, $X \subseteq \mathbb{R}^n$, $X_0 \subseteq X$, $X_\mathrm{u} \subseteq X$, and $D \subseteq \mathbb{R}^r$. In the safety verification, we only consider trajectories initialized in X_0 that are contained in the set X. We verify if there exists a trajectory that can reach an unsafe set X_u. The safety of a system Γ is defined as follows.

Definition 35 (Safety)**.** Let $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ be a dynamical system. A trajectory $\phi^{\bar{d}}_{X_0} : [0, T] \to \mathbb{R}^n$ with disturbance $\bar{d}$ is unsafe if there exists a time $t \in [0, T]$, such that $\phi^{\bar{d}}_{X_0}([0, t]) \cap X_\mathrm{u} \neq \emptyset$ and $\phi^{\bar{d}}_{X_0}([0, t]) \subseteq X$.

We say that the system Γ is safe if there are no unsafe trajectories.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{Z}(f)$ denotes the set

$$\mathcal{Z}(f) \equiv \{x \in \mathbb{R}^n | f(x) = 0\}. \tag{4.3}$$

The Lie derivative of a differentiable function $B : \mathbb{R}^n \to \mathbb{R}$ along $f : \mathbb{R}^{n+r} \to \mathbb{R}^n$ is denoted by

$$L_f B(x, d) \equiv \frac{\partial B}{\partial x}(x) f(x, d). \tag{4.4}$$

The safety of a system Γ can be verified using the following condition.

Proposition 14 (Strict barrier certificate [PJP07])*: Let $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ be given. If there exists a differentiable function $B : X \to \mathbb{R}$ satisfying*

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{4.5a}$$

$$B(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{4.5b}$$

$$L_f B(x, d) < 0 \qquad \forall (x, d) \in \mathcal{Z}(B) \times D. \tag{4.5c}$$

Then the system Γ is safe.

Proposition 14 states that a trajectory initialized within the zero sublevel set of a function B, cannot cross the zero level set $\mathcal{Z}(B)$, if B is decreasing (along system trajectories) on the zero level set. This setup is illustrated in Figure 4.1.

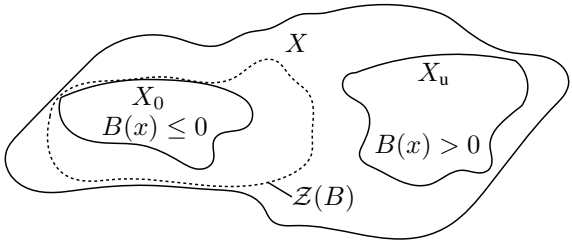


Figure 4.1: Illustration of a set X, which contains the initial set X_0 and the unsafe set X_{u}. The dashed line illustrates the zero level set of B.

The set of barrier certificates satisfying Proposition 14 is nonconvex, due to (4.5c). However, the following more conservative condition has a convex set of feasible barrier certificates. The convexity property becomes apparent in the computation of the barrier certificates.

Proposition 15 (Weak barrier certificate [PJP07, PJ04])*: Let* $\Gamma = (f, X, X_0, X_{\mathrm{u}}, D)$ *be given. If there exists a differentiable function* $B : X \to \mathbb{R}$ *satisfying*

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{4.6a}$$
$$B(x) > 0 \qquad \forall x \in X_{\mathrm{u}}, \text{ and} \tag{4.6b}$$
$$L_f B(x, d) \leq 0 \qquad \forall (x, d) \in X \times D. \tag{4.6c}$$

Then the system Γ *is safe.*

Proposition 15 states that a trajectory of a system initialized in a state within the zero sublevel set of a nonincreasing function (along system trajectories), cannot reach the complement of the zero sublevel set.

The difference between Proposition 14 and Proposition 15 is that (4.6c), in contrast to (4.5c), must hold for all states and all disturbances. Additionally, (4.5c) is a strict inequality constraint weathers (4.6c) is weak.

The subtle differences between Proposition 14 and Proposition 15 make a big difference for the computation of the barrier certificate B. Therefore, we only provide methods for computing barrier certificates using Proposition 15 in the rest of the chapter. Consult Paper D for details about computing barrier certificates using Proposition 14.

4.2 Compositional Safety Verification

The purpose of developing a compositional safety verification method is to improve the scalability of the safety verification. This is needed when polynomial systems are considered. A centralized method failed to verify the example provided in Paper D; however, safety verification was possible using the compositional method presented in this chapter.

The safety verification problem is posed as a compositional problem, by assuming that a dynamical system is given as an interconnection of subsystems. We discard all disturbances until Section 4.5, as they have no influence on the compositional properties

of the system; however, disturbances are included in Paper D. First, an interconnected system is defined.

Definition 36. Let $\Gamma = (f, X, X_0, X_\mathrm{u})$ be a dynamical system with

$$\dot{x} = f(x), \tag{4.7}$$

where $x \in \mathbb{R}^n$ is the state.

Let $k \in \mathbb{N}$ and $x = (x_1, \ldots, x_k)$. For $i = 1, \ldots, k$, let $x_i \in \mathbb{R}^{n_i}$, let $g_i : \mathbb{R}^{n-n_i} \to \mathbb{R}^{m_i}$ and $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$ be continuous maps, and let $q \equiv \sum_i q_i$. Let $X = X_1 \times \cdots \times X_k$, $X_0 = X_{0,1} \times \cdots \times X_{0,k}$, and $X_\mathrm{u} = X_{\mathrm{u},1} \times \cdots \times X_{\mathrm{u},k}$. We say that the system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\})$ with

$$\Sigma_i : \begin{cases} \dot{x}_i &= f_i(x_i, g_i(\hat{x}_i)), \\ y_i &= h_i(x_i) \end{cases} \tag{4.8}$$

for $i = 1, \ldots, k$ is an interconnected system of $f(x)$ if

$$f(x) = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{4.9}$$

for all $x \in X$ and there exist maps $e_i : \mathbb{R}^{q-q_i} \to \mathbb{R}^{m_i}$ such that

$$g_i = e_i \circ \hat{h}_i. \tag{4.10}$$

An interconnected system is given by a collection of subsystems Σ_i with interconnection inputs $g_i(\hat{x}_i)$ and outputs $h_i(x_i)$ (Recall that $\hat{x}_i \equiv (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$). The interconnection of the subsystems is given by e_i.

To clarify the compositional setup, we initially consider a system from [TPM09] consisting of three interconnected subsystems, given by (4.8). The interconnection of the three subsystems is shown in Figure 4.2.

Let each subsystem have state $x_i \in X_i \subseteq \mathbb{R}^{n_i}$ and output $y_i \in \mathbb{R}^{q_i}$ given by the map $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$. Note that the interconnection of the three subsystems is defined by $e_i : \mathbb{R}^{q-q_i} \to \mathbb{R}^{m_i}$ that projects the possible inputs on the actual inputs of subsystem i. From Figure 4.2, it is seen that

$$e_1 : (y_2^1, y_2^2, y_3) \mapsto (y_2^1, y_3), \tag{4.11a}$$

$$e_2 : (y_1, y_3) \mapsto y_1, \tag{4.11b}$$

$$e_3 : (y_1, y_2^1, y_2^2) \mapsto y_2^2. \tag{4.11c}$$

Note that the map g_i gives the inputs to subsystem i and that the map h_i gives the outputs of subsystem i.

The interconnected system induces a graph structure, without self-loops and with only one edge from one vertex to another. The graph can be described by an adjacency matrix

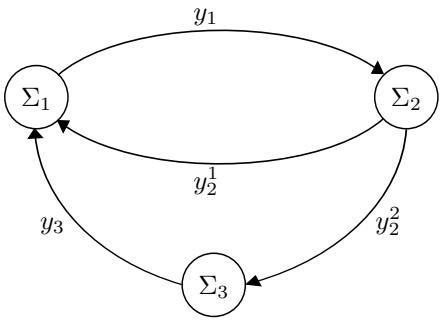


Figure 4.2: Interconnection of three subsystems $\Sigma_1, \Sigma_2, \Sigma_3$.

$E \in \mathbb{R}^k \times \mathbb{R}^k$, where $E(i,j) = 1$ if there is an edge between subsystem i and j, with the head at subsystem i and the tail at subsystem j. Note that the i^{th} row of E can be derived from e_i. For the graph in Figure 4.2 the adjacency matrix is

$$E = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.12}$$

In the following, we present three lemmas leading to the compositional safety verification shown in Corollary 4. The lemmas show how the inequality constraints on the barrier function and its derivative in Proposition 15 can be decomposed into separate constraints for each subsystem in addition to coupling constraints.

The following lemma shows how (4.6a) and (4.6b) can be decomposed.

Lemma 3. *Let $k \in \mathbb{N}$. For $i = 1, \ldots, k$, let $n_i \in \mathbb{N}$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ be a continuous function, and $X_i \subseteq \mathbb{R}^{n_i}$ be compact. There exist constants $c_i \in \mathbb{R}$ such that*

$$f_i(x_i) - c_i \leq 0 \quad \forall x_i \in X_i \text{ and} \tag{4.13a}$$

$$\sum_i c_i \leq 0 \tag{4.13b}$$

if and only if

$$\sum_i f_i(x_i) \leq 0 \quad \forall x_i \in X_i. \tag{4.14}$$

Proof. See Paper E on page 196. □

Lemma 3 shows that an inequality (4.14) in n variables is equivalent to k inequalities in n_i variables and an inequality constraint involving only constants. This equivalence is obtained by choosing c_i as the supremum of $f_i(x_i)$.

We put the following assumption on the output maps of the subsystems, to allow a reduction in the number of coupling variables in the decomposition of (4.6c).

Assumption 1*: Let Dh_i be the differential of h_i. For $i = 1, \ldots, k$*

$$Dh_i(x_i) \tag{4.15}$$

has constant rank.

The assumption guarantees that an output cannot occasionally "disappear".

The following lemma gives the reduction of coupling variables and relies on a coordinate transformation, which can be generated due to Assumption 1.

Lemma 4. *Let $\gamma : \mathbb{R}^n \to \mathbb{R}$ be a continuous function and let $h : \mathbb{R}^n \to \mathbb{R}^q$ be a smooth map such that Dh has constant rank k. Then there is a smooth map $\bar{h} : \mathbb{R}^n \to \mathbb{R}^{n-k}$ such that $D\bar{h}$ has constant rank $n-k$, and a continuous function $\tilde{\gamma} : \mathbb{R}^{q+(n-k)} \to \mathbb{R}$ such that*

$$\gamma(x) = \tilde{\gamma}(h(x), \bar{h}(x)) \qquad \forall x \in \mathbb{R}^n. \tag{4.16}$$

Proof. See Paper E on page 197. □

Lemma 4 allows the coupling constraint associated with (4.6c) in the following lemma, to be a function of only inputs and outputs of a subsystem.

Lemma 5. *Let $k \in \mathbb{N}$. For $i \in \{1, \ldots, k\}$, let*

- *$m_i, n_i, q_i \in \mathbb{N}$ and define $q \equiv \sum_i q_i$,*
- *$V_i \subseteq \mathbb{R}^{n_i+(n-n_i)}$ be compact,*
- *$f_i : \mathbb{R}^{n_i+m_i} \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R}^{n-n_i} \to \mathbb{R}^{m_i}$ be continuous maps,*
- *$\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ be a continuous function,*
- *$h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$ be a smooth map such that Dh_i has constant rank r_i.*

There exist continuous functions $\gamma_i : \mathbb{R}^{q_i+m_i} \to \mathbb{R}$ such that for all $(x_i, \hat{x}_i) \in V_i \subseteq \mathbb{R}^{n_i} \times \mathbb{R}^{n-n_i}$

$$\varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)), \text{ and} \tag{4.17a}$$

$$\sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \tag{4.17b}$$

if and only if

$$\sum_i \varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in V_i. \tag{4.18}$$

Proof. See Paper E on page 198. □

Notice the importance of γ_i only being a function of the inputs and outputs of subsystem i, in oppose to being a function of its entire state vector. This implies that the dimension of the coupling is drastically reduced if the number of interconnection variables is small compared to the number of states. Furthermore, it is important to note that

γ_i is continuous, as it enables γ_i to be approximated arbitrarily close by polynomials on a compact set [Sto48]. This is favorable, as polynomial inequality and equality constraints can be solved algorithmically by use of sum of squares programming [Par03].

In the following, we provide the compositional safety condition by rewriting Proposition 15 using Lemma 3 and Lemma 5.

Corollary 4: Let $k \in \mathbb{N}$ and let $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\})$ be a family of interconnected systems, with f_i, g_i, and h_i defined in Lemma 4. If there exist differentiable functions $B_i : X_i \to \mathbb{R}$, constants $\alpha_i, \beta_i \in \mathbb{R}$, and continuous functions $\gamma_i : \mathbb{R}^{q_i+m_i} \to \mathbb{R}$ for $i = 1, \ldots, k$ such that

$$B_i(x_i) + \alpha_i \leq 0 \qquad \forall x_i \in X_{0,i}, \tag{4.19a}$$

$$B_i(x_i) - \beta_i > 0 \qquad \forall x_i \in X_{\mathrm{u},i}, \tag{4.19b}$$

$$L_{f_i} B_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \qquad \forall (x_i, \hat{x}_i) \in X_i \times \hat{X}_i, \tag{4.19c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in X_i \times \hat{X}_i. \tag{4.19d}$$

Then the system Γ is safe.

It is seen that verification problem is decomposed into individual constraints for the subsystems given by (4.19a)-(4.19c) and coupling constraints given by (4.19d). In the next section, we show how to compute the compositional barrier certificates of Corollary 4.

4.3 Computation of Compositional Barrier Certificates

In this section, we show how to compute barrier certificates from the conditions set up in Corollary 4.

Remark 8: Any desired computational method can be applied to find the barrier certificates, and different methods can be applied on different subproblems for the compositional condition in Corollary 4. This is beneficial if some subsystems are linear and others are polynomial.

To demonstrate the computation of barrier certificates, we show how to compute the barrier certificates using sum of squares programming. Details on the computation of barrier certificates using linear programming are found in Paper D.

In this section, vector fields are assumed to be polynomial to allow their computation in a tool such as MATLAB. Furthermore, we parameterize the barrier certificates as polynomials, and require the invariant, initial, and unsafe sets to be semialgebraic. Initially, we set up some notation about polynomials.

Definition 37 (Polynomial [Par03])**.** A polynomial p in n variables $x_1, \ldots, x_n$ is a finite linear combination of monomials

$$p(x) = \sum_\alpha c_\alpha x^\alpha = \sum_\alpha c_\alpha x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}, \tag{4.20}$$

where $c_\alpha \in \mathbb{R}$ and the sum is over a finite number of n-tuples $\alpha = [\alpha_1, \ldots, \alpha_n]$ with $\alpha_i \geq 0$.

The total degree of a monomial x^α is $\alpha_1 + \cdots + \alpha_n$. Additionally, the total degree of a polynomial is equal to the highest degree of its component monomials. The degree of a polynomial p is denoted by $\deg(p)$.

We only consider polynomials with real valued variables, and denote the polynomial ring in n variables $\mathbb{R}[x_1, \ldots, x_n]$ by $\mathbb{R}[\underline{x}]$. Recall that a map $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be polynomial if its coordinate functions are polynomials, i.e., $f_i \in \mathbb{R}[\underline{x}]$ for $i = 1, \ldots, m$; hence, $f \in \mathbb{R}^m[\underline{x}]$.

Sum of squares polynomials are used in the generation of safety certificates and are explained in the following, based on [Par03].

Definition 38. A polynomial $p \in \mathbb{R}[\underline{x}]$ is called sum of squares (SOS) if

$$p = \sum_{i=1}^{k} p_i^2 \tag{4.21}$$

for some polynomials $p_i \in \mathbb{R}[\underline{x}]$ with $i = 1, \ldots, k$.

The set of sum of squares polynomials is a subset of nonnegative polynomials [Par03], which can be treated using semidefinite programming, as described below. We denote the set of sum of squares polynomials in n variables by Σ_n.

The existence of a sum of squares decomposition of a polynomial $p \in \mathbb{R}[\underline{x}]$, with $d = \deg(p)$, can be expressed as a semidefinite programming feasibility problem. Therefore, the formulation of a problem as sum of squares makes the problem computationally tractable; however, the number of decision variables in the program is

$$N = \binom{n + 2d}{2d} = \frac{(n + 2d)!}{2d!n!}. \tag{4.22}$$

In the search for sum of squares polynomials, it is exploited that the existence of a SOS decomposition of a polynomial p is equivalent to the existence of a positive semidefinite matrix $Q = Q^{\mathrm{T}} \succeq 0$ such that

$$p = Z^{\mathrm{T}} Q Z, \tag{4.23}$$

where Z is a vector of monomials of degree less than or equal to half the degree of p.

Let $k, l \in \mathbb{N}$, let $\alpha_{i,j} \in \mathbb{R}[\underline{x}]$ for $(i, j) \in \{1, \ldots, l\} \times \{1, \ldots, k\}$, and $w_j \in \mathbb{R}$. An SOS programming problem is

$$\underset{(c_1, \ldots, c_k) \in \mathbb{R}^k}{\text{minimize}} \sum_{j=1}^{k} w_j c_j \text{ subject to} \tag{4.24a}$$

$$\alpha_{i,0} + \sum_{j=1}^{k} \alpha_{i,j} c_j \in \Sigma_n \forall i = 1, \ldots, l. \tag{4.24b}$$

It is seen that an SOS programming problem is a minimization of a linear cost, subject to SOS feasibility constraints.

The interconnected system can be formulated as one system, but this would increase the number of decision variables involved in the safety verification, compared to the proposed compositional approach. This is an important issue when working with SOS optimization, and is apparent from (4.22).

To compute barrier certificates using sum of squares programming, we describe the invariant, initial, and unsafe sets as semialgebraic sets, i.e., they are given by polynomial inequalities. To ease the notation, we denote the interconnection input $g_i(\hat{x}_i)$ by u_i, and the output $h(x_i)$ by y_i.

Let $g_{X_i} : \mathbb{R}^{n_i} \to \mathbb{R}^{k_{X_i}}$, $g_{X_{0,i}} : \mathbb{R}^{n_i} \to \mathbb{R}^{k_{X_{0,i}}}$, and $g_{X_{\mathrm{u},i}} : \mathbb{R}^{n_i} \to \mathbb{R}^{k_{X_{\mathrm{u},i}}}$, and $g_{U_i} : \mathbb{R}^{m_i} \to \mathbb{R}^{k_{U_i}}$ for some $k_{X_i}, k_{X_{0,i}}, k_{X_{\mathrm{u},i}}, k_{U_i} \in \mathbb{N}$ be given as vectors of polynomials. Then

$$X_i \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_i}(x_i) \geq 0\}, \tag{4.25a}$$
$$X_{0,i} \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_{0,i}}(x_i) \geq 0\}, \tag{4.25b}$$
$$X_{\mathrm{u},i} \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_{\mathrm{u},i}}(x_i) \geq 0\}, \tag{4.25c}$$
$$U_i \equiv \{u_i \in \mathbb{R}^{m_i} | g_{U_i}(u_i) \geq 0\}, \tag{4.25d}$$

where the inequalities in (4.25) are satisfied *entry-wise*.

In the computation of barrier certificates, the following generalization of the $\mathcal{S}$-procedure is used [BGFB94].

Lemma 6. *Let V be a subset of $X \subseteq \mathbb{R}^n$. Let $f \in \mathbb{R}[\underline{x}]$ and $g \in \mathbb{R}^k[\underline{x}]$. Suppose $g(x) \geq 0$ (element-wise) for any $x \in V$. If*

1. *$\lambda \in \Sigma_n^k$ and*
2. *$f - \lambda^{\mathrm{T}} g \in \Sigma_n$.*

Then $f(x) \geq 0$ for all $x \in V$.

Corollary 4 is written in terms of SOS by the use of (4.25) and Lemma 6.

Corollary 5: Let $k \in \mathbb{N}$, the polynomials g_* shown in (4.25), and the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\})$ be given, and let $\epsilon_1 > 0$. If there exist $B \in \mathbb{R}[\underline{x}_i]$, $\alpha_i \in \mathbb{R}$, $\beta_i \in \mathbb{R}$, $\gamma_i \in \mathbb{R}[\underline{y}_i, \underline{u}_i]$, $\lambda_{X_{0,i}} \in \Sigma_{n_i}^{k_{X_{0,i}}}$, $\lambda_{X_{\mathrm{u},i}} \in \Sigma_{n_i}^{k_{X_{\mathrm{u},i}}}$, $\lambda_{X_i} \in \Sigma_{n_i+m_i+q_i}^{k_{X_i}}$, and $\lambda_{U_i} \in \Sigma_{n_i+m_i+q_i}^{k_{U_i}}$ such that

$$-B_i - \lambda_{X_{0,i}}^{\mathrm{T}} g_{X_{0,i}} - \alpha_i \in \Sigma_{n_i}, \tag{4.26a}$$
$$B_i - \epsilon_1 - \lambda_{X_{\mathrm{u},i}}^{\mathrm{T}} g_{X_{\mathrm{u},i}} - \beta_i \in \Sigma_{n_i}, \text{ and} \tag{4.26b}$$
$$-L_{f_i} B_i + \gamma_i - \lambda_{X_i}^{\mathrm{T}} g_{X_i} - \lambda_{U_i}^{\mathrm{T}} g_{U_i} \in \Sigma_{n_i+m_i} \tag{4.26c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \text{ and } -\sum_i \gamma_i \in \Sigma_{n_i+m_i}. \tag{4.26d}$$

Then the system Γ is safe.

Corollary 5 can be solved directly; however, it is clear from (4.26d) that the subproblems are not yet decomposed. The subproblems are decomposed using dual decomposition [BXMM08] in the following.

It is observed that γ_i should have a particular structure to ease computations. Therefore, γ_i is chosen as

$$\gamma_i = \begin{bmatrix} Z(y_i) \\ Z(u_i) \end{bmatrix}^{\mathrm{T}} P_i \begin{bmatrix} Z(y_i) \\ Z(u_i) \end{bmatrix}, \tag{4.27}$$

where $Z(y_i)$ and $Z(u_i)$ are vectors of monomials in y_i respectively u_i, and P_i is a diagonal matrix. For convenience, we let $\bar{\gamma}_i$ be a vector containing the diagonal elements of P_i.

Let $\lambda \equiv (\lambda_1,\, \lambda_2,\, \lambda_3)$. The dual function is

$$\varphi(\lambda) = \sum_i \varphi_i(\lambda), \tag{4.28}$$

where

$$\varphi_i(\lambda) \equiv \inf_{\alpha_i, \beta_i, \bar{\gamma}_i} -\lambda_1 \alpha_i - \lambda_2 \beta_i + \lambda_3^{\mathrm{T}} \bar{\gamma}_i \tag{4.29}$$

subject to

$$-B_i - \lambda_{X_{0,i}}^{\mathrm{T}} g_{X_{0,i}} - \alpha_i \in \Sigma_{n_i}, \tag{4.30a}$$

$$B_i - \epsilon_1 - \lambda_{X_{\mathrm{u},i}}^{\mathrm{T}} g_{X_{\mathrm{u},i}} - \beta_i \in \Sigma_{n_i}, \text{ and} \tag{4.30b}$$

$$-L_{f_i} B_i + \gamma_i - \lambda_{U_i}^{\mathrm{T}} g_{U_i} \in \Sigma_{n_i + m_i}. \tag{4.30c}$$

Remark that λ_3 is a vector. The dual problem becomes

$$\sup_{\lambda \geq 0} \sum_i \varphi_i(\lambda). \tag{4.31}$$

In the following, we explain how the subgradient algorithm can be used to solve the previous optimization problem [SKR85]. Let $\alpha_i^*(\lambda)$ be the optimal value of α_i for a given λ. Then the gradients of $\varphi_1(\lambda), \ldots, \varphi_k(\lambda)$ are

$$g_i(\lambda) = \begin{bmatrix} \alpha_i^*(\lambda) & \beta_i^*(\lambda) & \gamma_i^*(\lambda) \end{bmatrix}. \tag{4.32}$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, and let $x, y \in \mathbb{R}^n$. Then any vector $g \in \mathbb{R}^n$ that satisfies

$$f(y) \geq f(x) + g^{\mathrm{T}}(y - x) \tag{4.33}$$

is called a subgradient at x. From (4.32) and (4.33), we get for all $\mu \equiv (\mu_1, \mu_2, \mu_3)$ and $i = 1, \ldots, k$

$$\varphi_i(\mu) \geq \varphi_i(\lambda) + g_i(\mu - \lambda). \tag{4.34}$$

The function to be maximized is $\varphi(\lambda) = \sum_i \varphi_i(\lambda)$, which has a gradient $g(\lambda^{(k)}) = \sum_i g_i(\lambda^{(k)})$. The vector of multipliers is updated according to

$$\lambda^{(k+1)} = \lambda^{(k)} - \Delta_k g^{\mathrm{T}} \left(\lambda^{(k)} \right), \tag{4.35}$$

with

$$\Delta_k = \frac{a}{b+k}, \tag{4.36}$$

where $a > 0$ and $b \geq 0$. A diminishing step size Δ_k is chosen to guarantee that the algorithm converges to the optimal value, see [Pol77].

It is seen that if $\sum_i \alpha_i \geq 0$ is violated, then $\lambda_1^{(k+1)} > \lambda_1^{(k)}$, as the first element of $g(\lambda^{(k)})$ is negative. This puts a larger penalty on the violation of the constraint through the dual variable λ_1.

4.4 Existence of Compositional Barrier Certificates

In this section, we show that Proposition 15 and Corollary 4 are equivalent, if the barrier function is assumed to be additively separable. This equivalence important to identify, to determine when it is reasonable to use the compositional method.

Definition 39. Let $k \in \mathbb{N}$ and $i = 1, \ldots, k$. We say that a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is additively separable in $x = (x_1, \ldots, x_k)$ if there exist functions $\varphi_i : \pi_i(\mathbb{R}^n) \to \mathbb{R}$, where π_i is a projection that takes $(x_1, \ldots, x_k)$ to x_i such that

$$\varphi(x) = \sum_i \varphi_i(x_i) \qquad \forall x \in \mathbb{R}^n, \tag{4.37}$$

where $x_i \in \mathbb{R}^{n_i}$ and $n = \sum_i n_i$.

We can now state when Proposition 15 and Corollary 4 are equivalent. Recall that $\hat{x}_i \equiv (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. Similarly, given a sequence of maps $(h_1, \ldots, h_k)$, we define $\hat{h}_i \equiv (h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_k)$.

Theorem 7: Let $k \in \mathbb{N}$, and let

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{4.38}$$

be an interconnected system of $f(x)$.

There exists an additively separable continuous function $B : \mathbb{R}^n \to \mathbb{R}$ such that

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{4.39a}$$
$$B(x) > 0 \qquad \forall x \in X_{\mathrm{u}}, \text{ and} \tag{4.39b}$$
$$L_f B(x) \leq 0 \qquad \forall x \in X \tag{4.39c}$$

if and only if for $i = 1, \ldots, k$ there exist continuous functions $\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and $\gamma_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$ and constants $\alpha_i, \beta_i \in \mathbb{R}$ such that

$$B_i(x_i) + \alpha_i \leq 0 \qquad \forall x \in X_{0,i}, \tag{4.40a}$$
$$B_i(x_i) - \beta_i > 0 \qquad \forall x \in X_{\mathrm{u},i}, \tag{4.40b}$$
$$L_{f_i} B_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \qquad \forall x \in X \tag{4.40c}$$

and

$$\sum_i \alpha_i \geq 0, \tag{4.40d}$$

$$\sum_i \beta_i \geq 0, \text{ and} \tag{4.40e}$$

$$\sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall x \in X. \tag{4.40f}$$

Proof. See Paper E on page 201. □

Remark that to generate the additively separable barrier functions, one should only use the sum of bases in x_i for $i \in \{1, \ldots, k\}$.

Refined Compositional Analysis

To alleviate potential issues with the compositional method, we propose another compositional condition for safety, which at the cost of more coupling variables successfully handles a greater class of systems. The idea is to let each B_i depend on both x_i and $g_i(\hat{x}_i)$.

To simplify the notation of the problem, we define the set of neighbors for subsystem i, as the set of subsystems, which has an output that is an input to subsystem i. The set of neighbors is defined from the adjacency matrix E, see (4.12), describing the interconnection of the subsystems. We say that the neighbors of subsystem i have the following indices

$$\mathcal{N}_i = \{j \in \{1, \ldots, k\} | E(i, j) = 1\}. \tag{4.41}$$

Define $\bar{\mathcal{N}}_i \equiv \mathcal{N}_i \cup \{i\}$. The complement of $\bar{\mathcal{N}}_i$ is given as

$$\bar{\mathcal{N}}_i^{\mathrm{c}} = \{1, \ldots, k\} \backslash \bar{\mathcal{N}}_i. \tag{4.42}$$

Let $z = (z_1, \ldots, z_k)$ and $A \subseteq \{1, \ldots, k\}$, then we define $\hat{z}_A \equiv \{z_i | i \in \{1, \ldots, k\} \backslash A\}$ and $z_A \equiv \sum_{i \in A} z_i$.

Now, we can state the refined safety condition as follows.

Proposition 16*: Let $k \in \mathbb{N}$, and let*

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{4.43}$$

be an interconnected system of $f(x)$.

There exists a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ given by

$$B(x) = \sum_i B_i(x_i, g_i(\hat{x}_i)) \qquad \forall x \in \mathbb{R}^n \tag{4.44}$$

such that

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{4.45a}$$
$$B(x) > 0 \qquad \forall x \in X_{\mathrm{u}}, \text{ and} \tag{4.45b}$$
$$L_f B(x) \leq 0 \qquad \forall x \in X \tag{4.45c}$$

if and only if for $i \in \{1, \ldots, k\}$ *there exist continuous functions* $\gamma_i : \mathbb{R}^{q_{\bar{\mathcal{N}}_i} + m_{\mathcal{N}_i}} \to \mathbb{R}$, $\alpha_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$, *and* $\beta_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$ *such that*

$$B_i(x_i, g_i(\hat{x}_i)) + \alpha_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall x \in X_0, \tag{4.46a}$$
$$B_i(x_i, g_i(\hat{x}_i)) - \beta_i(h_i(x_i), g_i(\hat{x}_i)) > 0 \qquad \forall x \in X_{\mathrm{u}}, \tag{4.46b}$$
$$\sum_{j \in \bar{\mathcal{N}}_i} \frac{\partial B_i}{\partial x_j}(x_i, g_i(\hat{x}_i)) f_j(x_j, g_j(\hat{x}_j)) \leq \gamma_i(\hat{x}_{\mathcal{N}_i^{\mathrm{c}}}, \hat{g}_{\mathcal{N}_i^{\mathrm{c}}}(\hat{x}_{\mathcal{N}_i^{\mathrm{c}}})) \ \ \forall x \in X \tag{4.46c}$$

and

$$\begin{aligned} &\sum_i \alpha_i(h_i(x_i), g_i(\hat{x}_i)) \geq 0 \qquad \forall x \in X, \\ &\sum_i \beta_i(h_i(x_i), g_i(\hat{x}_i)) \geq 0 \qquad \forall x \in X, \\ &\sum_i \gamma_i(\hat{x}_{\mathcal{N}_i^{\mathrm{c}}}, \hat{g}_{\mathcal{N}_i^{\mathrm{c}}}(\hat{x}_{\mathcal{N}_i^{\mathrm{c}}})) \leq 0 \qquad \forall x \in X. \end{aligned} \tag{4.46d}$$

Proof. See Paper E on page 203. □

The seemingly subtle change of B_i has a great impact on the number of coupling variables involved in the generation of the barrier certificate. Therefore, one should only include $g_i(\hat{x}_i)$ in B_i if it is really necessary. Remark that a subset of functions B_i may be dependent of $g_i(\hat{x}_i)$, while others may only depend on x_i.

This finalized the presentation of the compositional verification method. The next section presents a controller design method based on barrier certificates.

4.5 Design of Safe Controllers

The purpose of this section is to present the developed methodology for designing safe controllers. The content of this section originates from Paper F.

We consider a system of ordinary differential equations that are affine in control and disturbance

$$\dot{x} = f(x) + g(x)u + h(x)d, \tag{4.47}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $d \in D \subseteq \mathbb{R}^r$ is the disturbance input. A control system given by $\Gamma = (f, g, h, X, X_0, X_{\mathrm{u}}, D)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$, and $h : \mathbb{R}^n \to \mathbb{R}^{n \times r}$ are continuous, $X \subseteq \mathbb{R}^n$, $X_0 \subseteq X$, $X_{\mathrm{u}} \subseteq X$, and $D \subseteq \mathbb{R}^r$ is convex. The system is controlled via the continuous map $k : \mathbb{R}^n \to \mathbb{R}^m$ defining the closed-loop behavior

$$f_{\mathrm{cl}} : x \mapsto f(x) + g(x)k(x). \tag{4.48}$$

The closed-loop system is denoted by $\Gamma_{\text{cl}} = (f_{\text{cl}}, h, X, X_0, X_{\text{u}}, D)$. For a measurable and essentially bounded disturbance function $\bar{d} : \mathbb{R}_{\geq 0} \to D$, we denote the solution of the Cauchy problem for the closed-loop system with $x(0) = x_0$ on an interval $[0, T]$ by $\phi^{\bar{d}}_{x_0}$, i.e.,

$$\frac{d\phi^{\bar{d}}_{x_0}(t)}{dt} = f_{\text{cl}}\left(\phi^{\bar{d}}_{x_0}(t)\right) + h\left(\phi^{\bar{d}}_{x_0}(t)\right)\bar{d}(t) \tag{4.49}$$

for almost all $t \in [0, T]$. We denote the set of solutions from all initial conditions x_0 in X_0 by $\phi^{\bar{d}}_{X_0}$.

The safety of a system $\Gamma_{\text{cl}} = (f_{\text{cl}}, h, X, X_0, X_{\text{u}}, D)$ is defined as shown in Definition 35.

The method addresses the following problem.

Problem 5: Given a system $\Gamma = (f, g, h, X, X_0, X_{\text{u}}, D)$, design a control law $k : \mathbb{R}^n \to \mathbb{R}^m$ such that the system $\Gamma_{\text{cl}} = (f_{\text{cl}}, h, X, X_0, X_{\text{u}}, D)$, where f_{cl} is given by (4.48)*, is safe.*

This problem is treated via the use of barrier certificates and is inspired by [WA07]. However, we redefine the control barrier function used in the controller design and allow unknown but bounded disturbances.

In the definition of a control barrier function, we denote the complement of X by X^{c}, and use the notion of contingent cone [Aub91]. Let K be a nonempty subset of a space X and let x belong to K. The contingent cone to K at x is the set

$$T_K(x) = \left\{ v \in X | \lim_{h \to 0+} \inf \frac{d_K(x + hv)}{h} = 0 \right\}, \tag{4.50}$$

where $d_K(y)$ denotes the distance of y to K, defined by

$$d_K = \inf_{z \in K} ||y - z||. \tag{4.51}$$

A control barrier function is defined as

Definition 40. Given a control system Γ. A continuously differentiable function $B : X \to \mathbb{R}$ satisfying

$$X_0 \subset B^{-1}((-\infty, 0]) \subset X_{\text{u}}^{\text{c}}, \text{ and} \tag{4.52a}$$

there exists $u \in \mathbb{R}^m$ such that for any $d \in D$ and $x \in B^{-1}(0)$

$$f(x) + g(x)u + h(x)d \in T_{B^{-1}((-\infty,0])}(x) \tag{4.52b}$$

is called a control barrier function.

Given a control barrier function, a control law must be found that ensures the safety of the system. A selection of such a control law is provided in the following, inspired by [Son89, WA07]. We denote the Lie derivative of B along f by $L_f B$.

Proposition 17*: Let* Γ *be a control system, let* B *be an associated proper control barrier function, and let* $L_g B(x) \neq 0$ *for* $x \in B^{-1}(0)$*. There exists a pair of real numbers* (γ_1, γ_2) *with* $0 < \gamma_1 < \gamma_2$ *such that the control*

$$k = -\xi(||b||)\frac{a + \alpha + \sqrt{(a+\alpha)^2 + \kappa^2 b^{\mathrm{T}} b}}{b^{\mathrm{T}} b} b, \tag{4.53}$$

where $a \equiv L_f B$, $b^{\mathrm{T}} \equiv L_g B$, $c^{\mathrm{T}} \equiv L_h B$, $\kappa > 0$,

$$\alpha(x) \equiv \sup_{d \in D} c^{\mathrm{T}}(x) d, \tag{4.54}$$

and $\xi : \mathbb{R} \to [0, 1]$ *defined by*

$$\omega(z) \equiv \begin{cases} 0 & \text{if } z \leq 0 \\ \exp(-1/z) & \text{if } z > 0 \end{cases} \tag{4.55}$$

$$\xi(z) \equiv \xi_{(\gamma_1, \gamma_2)}(z) \equiv \frac{\omega(z - \gamma_1)}{\omega(z - \gamma_1) + \omega(\gamma_2 - z)}, \tag{4.56}$$

is continuous and ensures safety for the closed-loop system Γ_{cl}.

Proposition 17 provides a state feedback controller that ensures safety of Γ_{cl}. The controller includes a bump function ξ that allows the controller to apply a zero control close to a desired reference, where the controller cannot guarantee that the system state gets closer to the reference.

The method is applied to control a model of a robot that should compensate for the movement of a heart in robotic heart surgery in Paper F.

4.6 Conclusion

In this chapter, a method for verifying a dynamical system decomposed into subsystems was presented. The method allows the verification of higher dimensional systems and is equivalent to the centralized method, when only additively separable functions are considered.

A controller design method that enforces the safety of a dynamical system was also presented. This method synthesizes state feedback controllers based on the barrier certificate method.

5 | Conclusions and Future Work

This chapter presents conclusions of the project. First, a summary of the contributions is provided, then conclusions are given, and finally suggestions for future research are presented.

5.1 Summary of Contributions

The contribution of this thesis is twofold: A method for verifying TCTL properties of continuous systems, and barrier certificate methods for ensuring safety of continuous systems.

The main contribution of Paper A-C is a methodology for abstracting continuous systems by timed automata. The abstraction method joins the sign based abstraction method with the introduction of time in [MB08], and most importantly, the method uses a subdivision of the state space generated by positively invariant sets. This allows the verification of TCTL properties of a continuous system via the verification of a complete abstraction given by a timed automaton. Additionally, sound abstractions allow the verification of universally quantified TCTL properties on positive normal form of continuous systems.

The key to the method is the definition of a slice via positively invariant sets that is the basic component of the subdivision. The cells of the subdivision are generated by the transversal intersection of a family of slices. This induces some favorable compositional properties of the abstraction. Additionally, robustness of the partition is ensured by only considering transversal intersection of sets generated by sublevel sets of functions with strictly negative Lie derivative along the vector field.

A necessary and sufficient condition is provided for subdivisioning functions that generate complete abstractions. This condition shows that the directional derivative of the subdivisioning function along the vector field must be a function of the subdivisioning function itself. This condition is paramount for generating abstractions, as it identifies *optimal* subdivisioning functions.

Finally, the abstraction method is modified to abstract mechanical systems, by using a slightly more liberal definition of subdivisioning function, defined based on a family of functions instead of a single function.

The main contribution of Paper D and Paper E is a compositional method for verifying the safety of continuous systems using barrier certificates. The method enables the verification of high dimensional systems, and is demonstrated on a shutdown procedure of a wind turbine. The compositional method is shown to be equivalent with the centralized method, when considering only additively separable barrier certificates. This gives

an indication of the applicability of the compositional method. If the method cannot be applied to a considered system, then refined conditions are provided that improves the applicability of the method at the cost of increased computational complexity.

Finally, Paper F presents a method for designing safe controllers for continuous systems, based to the barrier certificate method. The paper provides a new definition of a control barrier function that relates to the strict barrier certificates. Furthermore, it includes a constructive state feedback controller design method for systems with bounded disturbances. The method is applied to design a safe controller for motion compensation in beating-heart surgery.

5.2 Conclusions

At the outset of the thesis, the aim of the work was to allow formal verification of more complex specifications for more complex system models, in an attempt to move the frontier shown in Figure 1.1.

By designing a method for verifying TCTL properties of continuous systems, the verification of expressive specifications of continuous systems is possible. However, the verification relies on a subdivision of the state space that is very difficult to generate. Finding subdivisioning functions for generating sound abstractions is similar to finding a Lyapunov function for a system, and is simple for linear systems. Finding subdivisioning functions for generating complete abstractions is even more difficult. Therefore, it is most realistic to consider only sound abstractions, but optimize the partitioning functions, such that the over-approximation of the reachable set becomes as small as possible. This is possible using the proposed algorithms.

The abstraction method should be used to verify systems that are not possible to verify by simpler verification methods. Such examples are provided in Section 1.2, from which it is seen that relatively simple problems are difficult to verify.

The compositional barrier certificate method allows the safety verification of increasingly complex systems, more specifically, higher dimensional systems. The compositional method is favorable for systems that can be decomposed into lightly coupled subsystems. However, if this is not possible then the compositional method may fail to verify the safety of the system, as it can only find barrier functions in the space of additively separable functions. It is possible to consider more general barrier certificates, but this increases the computational complexity of the method considerably. Although the class of barrier functions may look restrictive, the applicability of the method was demonstrated in a wind turbine example.

The simplicity of the barrier certificate method also allowed the development of a controller design method. Since the conditions for barrier certificates are similar to Lyapunov stability conditions, the computational methods for stability are easily transferred to safety. The controller design method does not depend on numerical tools for the generation of the controller; hence, dimensionality of the system is in principle not an issue. However, the controller design method is based on state feedback, where output feedback is preferred.

5.3 Future Work

In this section, some suggested future work is presented. The proposals are both concrete problems to be solved and broader issues.

The work on abstracting continuous systems by timed automata lacks a software implementation. Therefore, such an implementation is relevant future work that would ease the application of the method to case studies. An implementation could also enable a comparison with other related verification methods to reveal for which problems the method is advantageous to use.

The generation of time information for abstractions is paramount for obtaining an accurate abstraction; however, this topic is not treated in depth. Therefore, existing results on calculating decay rates should be used in the abstraction procedure for the generation of sound abstractions.

It is desirable to generate complete abstractions; hence, an answer to the following problem is needed.

Problem 6*: Identify the class of vector fields for which there exists a complete abstraction.*

In relation to the generation of complete abstractions, the algorithm for generating subdivisioning functions assumes an affine relation between the subdivisioning function and its derivative along the vector field. This assumption should be relaxed to search for subdivisioning functions satisfying the sufficient and necessary condition provided in Theorem 3.

Finally, the abstraction method should be modified to allow the use of forms instead of functions to subdivide the state space.

To do a compositional safety verification of a dynamical system, it is crucial to do the subdivision into subsystems correctly. Therefore, an answer to the following problem is needed.

Problem 7*: Given a system* $\Gamma = (f, X, X_0, X_\mathrm{u})$ *and a positive number* $i \in \mathbb{N}$*. Find the optimal (with respect to safety verification) decomposition of* Γ *into* i *subsystems.*

In the safety verification of the shutdown procedure of a wind turbine presented in Paper E, it was actually desired to obtain the largest set of initial states for which the system was safe, i.e., an answer to the following problem.

Problem 8*: Given a system* $\Gamma = (f, X, X_\mathrm{u})$*. Find largest set of initial states* X_0 *such that the system* $\Gamma = (f, X, X_0, X_\mathrm{u})$ *is safe.*

In relation to the design of safe controllers, much work is needed. The proposed method is based on state feedback, but output feedback is often required (especially for the surgical application considered in Paper F). Therefore, results from controller design and the generation of control Lyapunov functions should be adopted for the design of safe controllers.

References

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 414–425, June 1990. doi:10.1109/LICS.1990.113766.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1016/0304-3975(94)90010-8.

[AHLP00] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000. doi:10.1109/5.871304.

[Alu] Rajeev Alur. Principles of embedded computation. Technical report, University of Pennsylvania.

[Alu11] Rajeev Alur. Formal verification of hybrid systems. In *Proceedings of the ninth ACM international conference on Embedded software*, pages 273–278, New York, NY, USA, 2011. ACM. doi:10.1145/2038642.2038685.

[Arn89] V.I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer-Verlag, 2. edition, 1989.

[ASB10] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010. doi:10.1016/j.nahs.2009.03.009.

[Aub91] Jean-Pierre Aubin. *Viability Theory*. Birkhäuser, 1991.

[BBE+07] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G.J. Pappas. Symbolic planning and control of robot motion. *IEEE Robotics Automation Magazine*, 14(1):61–70, March 2007. doi:10.1109/MRA.2007.339624.

[BCR98] Jacek Bochnak, Michel Coste, and Marie-Francoise Roy. *Real Algebraic Geometry*. Springer, 1998.

[BDFP00] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Are timed automata updatable? In E. Emerson and Aravinda Sistla, editors, *Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer Berlin / Heidelberg, 2000. doi:10.1007/10722167_35.

[BDL04] G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *International School on Formal Methods for the Design of Real-Time Systems*, volume 3185, pages 200–237, 2004.

[BG93] Stephen Boyd and Laurent El Ghaoui. Method of centers for minimizing generalized eigenvalues. *Linear Algebra and its Applications*, 188-189:63–111, 1993. doi:10.1016/0024-3795(93)90465-Z.

[BGFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *SIAM studies in applied mathematics*. SIAM, 1994. doi:10.1137/1.9781611970777.

[BH06] Calin Belta and Luc C.G.J.M. Habets. Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11):1749–1759, November 2006. doi:10.1109/TAC.2006.884957.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[Bla99] F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999. doi:10.1016/S0005-1098(99)00113-2.

[BPR98] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. Complexity of computing semi-algebraic descriptions of the connected components of a semi-algebraic set. In *Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, pages 25–29. ACM, 1998. doi:10.1145/281508.281533.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2nd edition, 2006. doi:10.1007/3-540-33099-2.

[Bre93] Glen E. Bredon. *Topology and Geometry*. Springer, 1993.

[Bro99] Mireille Broucke. A geometric approach to bisimulation and verification of hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 61–75. Springer Berlin / Heidelberg, 1999. doi:10.1007/3-540-48983-5_9.

[But05] Jeremy Butterfield. On symmetry and conserved quantities in classical mechanics. *arXiv:physics/0507192v1*, 2005.

[BXMM08] Stephen Boyd, Lin Xiao, Almir Mutapcic, and Jacob Mattingley. Notes on decomposition methods, 2008.

[CB99] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 1999.

[CK03] A. Chutinan and B.H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75, January 2003. doi:10.1109/TAC.2002.806655.

[DGH+11] Jerry Ding, Jeremy H. Gillula, Haomiao Huang, Michael P. Vitus, Wei Zhang, and Claire J. Tomlin. Hybrid systems in robotics. *IEEE Robotics & Automation Magazine*, 18(3):33–43, September 2011. doi:10.1109/MRA.2011.942113.

[DW61] George B. Dantzig and Philip Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29(4):767–778, 1961. Available from: http://www.jstor.org/stable/1911818.

[FGP06] Georgios Fainekos, Antoine Girard, and George J. Pappas. Temporal logic verification using simulation. In Eugene Asarin and Patricia Bouyer, editors, *Formal Modeling and Analysis of Timed Systems*, volume 4202 of *Lecture Notes in Computer Science*, pages 171–186. Springer Berlin / Heidelberg, 2006. doi:10.1007/11867340_13.

[FLGD+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer Berlin / Heidelberg, 2011. doi:10.1007/978-3-642-22110-1_30.

[Gir05] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer Berlin / Heidelberg, 2005. doi:10.1007/978-3-540-31954-2_19.

[Gir12] Antoine Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012. doi:10.1016/j.automatica.2012.02.037.

[GLZN09] Hervé Guéguen, Marie-Anne Lefebvre, Janan Zaytoon, and Othman Nasri. Safety verification and reachability analysis for hybrid systems. *Annual Reviews in Control*, 33(1):25–36, 2009. doi:10.1016/j.arcontrol.2009.03.002.

[GM12] Bernd Gärtner and Jiri Matousek. *Approximation Algorithms and Semidefinite Programming*. Springer, 2012. doi:10.1007/978-3-642-22015-9.

[Gol60] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, 1960.

[GP06] Antoine Girard and George Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 272–286. Springer Berlin / Heidelberg, 2006. doi:10.1007/11730637_22.

[GP07] Antoine Girard and George J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007. doi:10.1109/TAC.2007.895849.

[Has07] Brendan Hassett. *Algebraic Geometry*. Cambridge University Press, 2007.

[Hir76] Morris W. Hirsch. *Differential Topology*. Springer, Heidelberg, 1976.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998. doi:10.1006/jcss.1998.1581.

[JdM80] Jacob Palis Junior and Welington de Melo. *Geometric Theory of Dynamical Systems: An Introduction*. Springer, 1980.

[JFA+07] A. Julius, Georgios Fainekos, Madhukar Anand, Insup Lee, and George Pappas. Robust test generation and coverage for hybrid systems. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 329–342. Springer Berlin / Heidelberg, 2007. doi:10.1007/978-3-540-71493-4_27.

[JS98] Jorge V. Jose and Eugene J. Saletan. *Classical Dynamics: A Contemporary Approach*. Cambridge University Press, 1998.

[JT11] Manuel Mazo Jr. and Paulo Tabuada. Symbolic approximate time-optimal control. *Systems & Control Letters*, 60(4):256–263, 2011. doi:DOI:10.1016/j.sysconle.2011.02.002.

[KB06] Marius Kloetzer and Calin Belta. Reachability analysis of multi-affine systems. In João Hespanha and Ashish Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 348–362. Springer Berlin / Heidelberg, 2006. doi:10.1007/11730637_27.

[KB08] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, February 2008. doi:10.1109/TAC.2007.914952.

[KV97] A. B. Kurzhanski and I. Vályi. *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser Boston, 1997.

[KV00] Alexander Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 202–214. Springer Berlin / Heidelberg, 2000. doi:10.1007/3-540-46430-1_19.

[KV06] Alex A. Kurzhanskiy and Pravin Varaiya. Ellipsoidal toolbox (ET). In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1498–1503, December 2006. doi:10.1109/CDC.2006.377036.

[KV07] A.A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1):26–38, January 2007. doi:10.1109/TAC.2006.887900.

[KvdS10] Florian Kerber and Arjan van der Schaft. Compositional analysis for linear systems. *Systems & Control Letters*, 59(10):645–653, 2010. doi:10.1016/j.sysconle.2010.08.002.

[LCV10] B. Langerock, F. Cantrijn, and J. Vankerschaver. Routhian reduction for quasi-invariant Lagrangians. *Journal of Mathematical Physics*, 51(2), 2010. doi:10.1063/1.3277181.

[LPS00] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000. doi:10.1007/PL00009858.

[LTS99] John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999. doi:10.1016/S0005-1098(98)00193-9.

[Mar92] Jerrold E. Marsden. *Lectures on Mechanics*. Cambridge University Press, 1992.

[Mar08] Murray Marshall. *Positive Polynomials and Sums of Squares*, volume 146. American Mathematical Society, 2008.

[MB08] Oded Maler and Grégory Batt. Approximating continuous systems by timed automata. In *Formal Methods in Systems Biology*, volume 5054 of *Lecture Notes in Computer Science*, pages 77–89. Springer Berlin / Heidelberg, 2008. doi:10.1007/978-3-540-68413-8_6.

[MBT05] I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005. doi:10.1109/TAC.2005.851439.

[MDT10] Manuel Mazo, Anna Davitian, and Paulo Tabuada. Pessoa: A tool for embedded controller synthesis. In *Computer Aided Verification*, volume 6174 of *Lecture Notes in Computer Science*, pages 566–569. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-14295-6_49.

[Mey68] K. R. Meyer. Energy functions for Morse Smale systems. *American Journal of Mathematics*, 90(4):1031–1040, 1968. doi:10.2307/2373287.

[MHDK11] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis. Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 7(4):375–392, 2011. doi:10.1002/rcs.408.

[MT05] Ian Mitchell and Jeremy Templeton. A toolbox of hamilton-jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 480–494. Springer Berlin / Heidelberg, 2005. doi:10.1007/978-3-540-31954-2_31.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003. doi:10.1007/s10107-003-0387-5.

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 271–274. Springer Berlin / Heidelberg, 2004. doi:10.1007/978-3-540-24743-2_32.

[PJP07] S. Prajna, A. Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, August 2007. doi:10.1109/TAC.2007.902736.

[PLS00] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, jun 2000. doi:10.1109/9.863598.

[Pol77] B. T. Polyak. Subgradient methods: A survey of Soviet research. In *Proceedings of a IIASA Workshop*, volume 3 of *Nonsmooth Optimization*, pages 5–29, 1977.

[PPSP05] Stephen Prajna, Antonis Papachristodoulou, Peter Seiler, and Pablo A. Parrilo. SOSTOOLS and its control applications. In *Positive Polynomials in Control*, volume 312 of *Lecture Notes in Control and Information Sciences*, pages 273–292. Springer Berlin / Heidelberg, 2005. doi:10.1007/10997703_14.

[PR05a] Stephen Prajna and Anders Rantzer. On the necessity of barrier certificates. In *Proceedings of the 16th IFAC World Congress*, pages 526–531, 2005. doi:10.3182/20050703-6-CZ-1902.00743.

[PR05b] Stephen Prajna and Anders Rantzer. Primal–dual tests for safety and reachability. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 542–556. Springer Berlin / Heidelberg, 2005. doi:10.1007/978-3-540-31954-2_35.

[PR07] Stephen Prajna and Anders Rantzer. Convex programs for temporal verification of nonlinear dynamical systems. *SIAM Journal on Control and Optimization*, 46(3):999—-1021, 2007. doi:10.1137/050645178.

[Pra06] Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006. doi:10.1016/j.automatica.2005.08.007.

[PT09] G. Pola and P. Tabuada. Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization*, 48(2):719–733, 2009. doi:10.1137/070698580.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.

[SKR85] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[Son89] Eduardo D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989. doi:10.1016/0167-6911(89)90028-5.

[Sto48] M. H. Stone. The generalized Weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948. doi:10.2307/3029750.

[SW11] Christoffer Sloth and Rafael Wisniewski. Algorithmic approach to abstracting linear systems by timed automata. In *Proceedings of the 18th IFAC World Congress*, pages 4546–4551, Milano, Italy, August 2011. doi:10.3182/20110828-6-IT-1002.02568.

[Tab07] Paulo Tabuada. Symbolic models for control systems. *Acta Informatica*, 43:477–500, 2007. doi:10.1007/s00236-006-0036-6.

[Tab09] Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009. doi:10.1007/978-1-4419-0224-5.

[Tiw08] Ashish Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1):57—83, 2008. doi:10.1007/s10703-007-0044-3.

[TK02] Ashish Tiwari and Gaurav Khanna. Series of abstractions for hybrid automata. In *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 425–438. Springer Berlin / Heidelberg, 2002. doi:10.1007/3-540-45873-5_36.

[TK04] Ashish Tiwari and Gaurav Khanna. Nonlinear systems: Approximating reach sets. In Rajeev Alur and George Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 171–174. Springer Berlin / Heidelberg, 2004. doi:10.1007/978-3-540-24743-2_40.

[TLSS00] C.J. Tomlin, J. Lygeros, and S. Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, July 2000. doi:10.1109/5.871303.

[TMBO03] C.J. Tomlin, I. Mitchell, A.M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, July 2003. doi:10.1109/JPROC.2003.814621.

[TMG01] C. Tomlin, I. Mitchell, and R. Ghosh. Safety verification of conflict resolution manoeuvres. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):110–120, jun 2001. doi:10.1109/6979.928722.

[TP03] Paulo Tabuada and George J. Pappas. Model checking LTL over controllable linear systems is decidable. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 498–513. Springer Berlin / Heidelberg, 2003. doi:10.1007/3-540-36580-X_36.

[TPM09] U. Topcu, A.K. Packard, and R.M. Murray. Compositional stability analysis based on dual decomposition. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 1175–1180, December 2009. doi:10.1109/CDC.2009.5400309.

[Tu08] Loring W. Tu. *An Introduction to Manifolds*. Springer, 2008. doi:10.1007/978-1-4419-7400-6.

[WA07] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. In *Proceedings of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 462–467, 2007. doi:10.3182/20070822-3-ZA-2920.00076.

[Wis05] Rafael Wisniewski. *Flow Lines under Perturbations within Section Cones*. PhD thesis, Department of Mathematical Sciences, Aalborg University, 2005.

[WTM10] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon control for temporal logic specifications. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, HSCC '10, pages 101–110, New York, NY, USA, 2010. ACM. doi:10.1145/1755952.1755968.

Contributions

Paper A

Verification of Continuous Dynamical Systems by Timed Automata

Christoffer Sloth and Rafael Wisniewski

This paper was published in:
Formal Methods in System Design, 39(1): 47—82, August 2011

The layout has been revised

Abstract

This paper presents a method for abstracting continuous dynamical systems by timed automata. The abstraction is based on subdividing the state space of a dynamical system using positive invariant sets, which form cells that represent locations of a timed automaton. The abstraction is intended to enable formal verification of temporal properties of dynamical systems without simulating any system trajectory, which is currently not possible. Therefore, conditions for obtaining sound, complete, and refinable abstractions are set up.

The novelty of the method is the subdivision of the state space, which is generated utilizing sub-level sets of Lyapunov functions, as they are positive invariant sets. It is shown that this subdivision generates sound and complete abstractions. Furthermore, the complete abstractions can be composed of multiple timed automata, allowing parallelization of the verification process. The proposed abstraction is applied to two examples, which illustrate how sound and complete abstractions are generated and the type of specification we can check. Finally, an example shows how the compositionality of the abstraction can be used to analyze a high-dimensional system.

1 Introduction

The verification of properties such as safety is important for any system. Such verification involves reachability calculations or approximations. The reachable sets of continuous and hybrid systems are in general incomputable [1]. Therefore, much research effort has been spent on the approximation of especially reachable sets for continuous systems [2]. Yet, reachability is decidable for systems such as automata and timed automata; consequently, there exists a rich set of tools aimed at verifying properties of such systems. Therefore, abstracting dynamical systems by discrete systems would enable the verification of dynamical systems using the tools for discrete systems.

There are basically two methods for verifying continuous and hybrid systems. The first is to over-approximate the reachable states by convex sets as in [3, 4, 5]. The second method is to abstract the original system by a system of reduced complexity. Both methods rely on reach set computations, which according to [2] limits the capabilities of automatic analysis, due to their complexity. An abstraction method for continuous systems is presented in [6], and an abstraction method for hybrid systems is presented in [7]. The models used in these methods are called symbolic models if equivalence classes of states are used instead of individual states [8]; for more insight in symbolic dynamics see [9]. To reduce the computational effort, the verification process is often accomplished by first choosing a coarse subdivision and then refining it until the system can be verified. One such algorithm is proposed in [10], where piecewise affine systems with real eigenvalues are considered.

The goal of this paper is to abstract continuous systems by timed automata, since efficient tools such as UPPAAL can verify this type of models [11]. The verification of temporal requirements with a bounded time horizon is well known and studied in computer science, via checking if some model satisfies, e.g., a Timed Computation Tree Logic (TCTL) specification [12], but not in control theory. In control theory almost all requirements are related to convergence, i.e., system properties when time goes to infinity. This implies that the formal verification of temporal requirements of dynamical systems would replace the need for simulations to verify the transient behavior of the systems.

Hence, the proposed method has relevance for even the most simple dynamical system models.

The concept of the abstraction is inspired by [6], where slices are introduced to improve the accuracy of abstractions of continuous systems. In short, the aim of [6] is to abstract autonomous continuous systems by timed automata via subdividing their state spaces into cubes along the coordinate axis as shown in Figure 6.1. As a result, each cube is associated with a discrete location of the timed automaton. Slices are used in addition to the cells in the generation of the abstraction, to increase the precision of the abstraction. A slice is a collection of cells, as illustrated in Figure 6.1 by the collection of shaded cells.

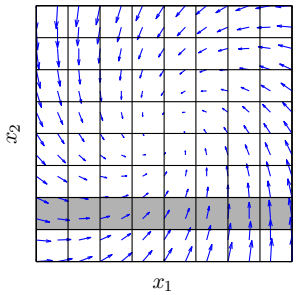


Figure 6.1: The figure illustrates a vector field of a dynamical system (blue arrows) and a subdivision of its state space into cubes (black squares). The shaded cubes represent a slice.

Previous abstraction techniques are based on over-approximations, where the upper bound on the error in general is unknown. This implies that the quality of the abstraction is unknown and that falsification of safety is not possible unless refinements or under-approximations are considered. However, this is not considered in this paper.

In this paper, which is an extension of [13], continuous systems are abstracted by timed automata by considering both cells and slices for generating the abstractions. A new subdivision is proposed, where the functions used for the subdivisioning are chosen in accordance with the vector field of the dynamical system. By this, we mean that the vector field cannot be tangent to a boundary of a cell. This requirement is not satisfied in the subdivision illustrated in Figure 6.1, as some of the blue arrows (illustration of the vector field) are tangent to the boundaries of cells. This approach is different from most previous work on abstractions of continuous systems; however, this may reduce the size of the symbolic models, which is currently the focus of other research [14]. Additionally, model checking can be done in a compositional manner using the proposed method, due to the use of slices. This may reduce the computational complexity for high-dimensional systems. Our objective is to show that by subdividing the state space in accordance with the vector field of the dynamical system, it is possible to prove safety of Morse-Smale systems using the proposed abstraction and additionally to falsify safety for linear systems. Remark that falsification of safety properties may also be possible by, e.g., refining the abstraction or doing analysis based on under-approximations. Furthermore, for linear systems, it is possible to calculate an a priori upper bound of the size of the over-approximation of the reachable set and reduce this upper bound to an arbitrary small

value, by refining the subdivision. Hence, we can obtain an abstraction with arbitrary precision of the reachable set. In conclusion, the following problem is formulated.

Problem 9: Given an autonomous dynamical system, find a subdivision of its state space, which allows over-approximation with arbitrary accuracy of its reachable set by a timed automaton.

To ease the flow of the article, some definitions and the proofs of the presented propositions are located in appendix.

This paper is organized as follows. Section 2 contains preliminary definitions utilized throughout the paper. Then the general idea of the abstraction is explained in Section 3 and Section 4. First, we determine how a subdivision of a state space can be generated by positive invariant sets, and then we show how a timed automaton is generated from this subdivision. In Section 5, properties of the generated timed automaton are derived. In Section 6, conditions for the subdivision are deduced, and LMI conditions for their satisfaction are provided. Afterwards, a method for synthesizing such a subdivision is proposed in Section 7. Examples are provided in Section 8; and Section 9 comprises conclusions.

Notation

The set $\{1, \ldots, k\}$ is denoted $\boldsymbol{k}$. B^A is the set of maps $A \to B$. The power set of A is denoted 2^A. Given a vector $a \in \mathbb{R}^n$, $a(j)$ denotes the j^{th} coordinate of a. Given a set A, the cardinality of the set is denoted $|A|$. We consider the Euclidean space $(\mathbb{R}^n, \langle , \rangle)$, where $\langle , \rangle$ is the scalar product. Whenever $f : X \to \mathbb{R}$ is a function and $a \in \mathbb{R}$, we write $f^{-1}(a)$ to shorten the notation of $f^{-1}(\{a\})$. $\mathfrak{X}^r(M)$ is the space of C^r vector fields.

2 Preliminaries

The purpose of this section is to provide definitions related to dynamical systems and timed automata. Especially, we give a detailed description of the considered class of dynamical systems.

Dynamical Systems

This subsection provides a definition of an autonomous dynamical system and its solution. This is followed by definitions of reachable set and Lyapunov function used for the subdivision. Finally, explanations of the considered class of systems are provided.

Definition 41 (Autonomous Dynamical System)**.** An autonomous dynamical system $\Gamma = (X, f)$, with state space $X \subseteq \mathbb{R}^n$ and $f : X \to \mathbb{R}^n$ a continuous map, has dynamics described by ordinary differential equations

$$\dot{x} = f(x). \tag{6.1}$$

Let $\phi_\Gamma : [0, \epsilon] \times X_0 \to X$, $\epsilon > 0$ be the flow map satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{6.2}$$

for all $t \in [0, \epsilon]$ and $\phi_\Gamma(0, x_0) = x_0$. In other words, $\phi_\Gamma(t, x_0)$ is the solution of (6.1), from an initial state $x_0 \in X_0 \subseteq X$ for $t \in [0, \epsilon]$.

It is assumed that (6.1) has a solution for each $x_0 \in X_0$ and for all time $t \in \mathbb{R}$, and that this solution is unique. Hence, the function f is continuous, locally Lipschitz, and has linear growth [15].

The reachable set of a dynamical system is defined in the following.

Definition 42 (Reachable set of Dynamical System)**.** The reachable states of a system Γ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1, t_2]$ is defined as

$$\text{Reach}_{[t_1,t_2]}(\Gamma, X_0) = \{x \in X |\ \exists t \in [t_1, t_2], \exists x_0 \in X_0, \text{ such that } x = \phi_\Gamma(t, x_0)\}. \tag{6.3}$$

We define Lyapunov function in the following, as these are used in the subdivisioning [16].

Definition 43 (Lyapunov Function)**.** Let X be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous and let $\text{Cr}(f)$ be the set of critical points of f. Then a real non-degenerate (see [17, p. 1]) differentiable function $\varphi : X \to \mathbb{R}$ is said to be a Lyapunov function for f if

$$p \text{ is a critical point of } f \Leftrightarrow p \text{ is a critical point of } \varphi$$

$$L_f\varphi(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f^j(x) \tag{6.4a}$$

$$L_f\varphi(x) = 0 \quad \forall x \in \text{Cr}(f) \tag{6.4b}$$

$$L_f\varphi(x) < 0 \quad \forall x \in X \backslash \text{Cr}(f) \tag{6.4c}$$

and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \text{Cr}(f)$, where

$$||L_f\varphi(x)|| \geq \alpha ||x - p||^2. \tag{6.5}$$

Notice that we only require the vector field to be transversal to the level curves of a Lyapunov function φ, i.e., $L_f\varphi(x) = \langle \nabla\varphi(x), f(x) \rangle < 0$ for all $x \in X \backslash \text{Cr}(f)$, and does not use Lyapunov functions in the usual sense, where the existence of a Lyapunov function implies stability, but uses a more general notion from [16]. Assume that a Lyapunov function φ is positive definite, then its sub-level sets generate positive invariant sets.

Definition 44 (Positive Invariant Set)**.** Given a system $\Gamma = (X, f)$, a set $\mathcal{X} \subseteq X$ is said to be positively invariant if for all $x_0 \in \mathcal{X}$ and for all $t \geq 0$

$$\phi_\Gamma(t, x_0) \in \mathcal{X}. \tag{6.6}$$

Considered Class of Systems

In this subsection we describe the considered class of systems and motivate why this class of systems is sufficient for this work.

The most general class of systems considered in this work is Morse-Smale systems, defined formally in Definition 72. However, some of the results only apply for linear systems. A linear system $\Gamma = (X, f)$ is a system with linear f, i.e.,

$$\dot{x} = Ax \tag{6.7}$$

where A is an $n \times n$ non-singular matrix.

To explain Morse-Smale systems we first explain topological equivalence of systems and structural stability of systems.

Two systems (or vector fields) are topological equivalent, see Definition 73, if there exists a correspondence between their solution trajectories, given by a continuous deformation (homeomorphism, i.e., a continuous bijection with continuous inverse). Hence, the exists a homeomorphism h such that the solution of a vector field η from some initial state can be described by a solution of a topologically equivalent vector field ξ as $h \circ \phi_\xi(\mathbb{R}, x_0) = \phi_\eta(\mathbb{R}, h(x_0))$.

Next, we define structural stability of a vector field, which is an important property of Morse-Smale systems.

Definition 45 (Structurally Stable Vector Field [18])**.** A vector field $\xi \in \mathfrak{X}^r(M)$ is structurally stable if there exists a neighborhood V of ξ in $\mathfrak{X}^r(M)$ such that every $\eta \in V$ is topologically equivalent to ξ.

A vector field is structurally stable if its quantitative behavior does not change after the vector field has been slightly perturbed. In the following, we provide some intuition in structurally stable systems, by showing an example of a system that is not structurally stable.

Consider the linear system with purely complex eigenvalues $\{-i, i\}$, i.e., the real parts of the eigenvalues are zero

$$\dot{x} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} x. \tag{6.8}$$

Two trajectories of the system are drawn in the left subplot of Figure 6.2. If we slightly perturb the system (This perturbation is given by a smooth map, see Theorem 2.1 in [19]), the eigenvalues of the system may become positive (trajectory in the middle subplot) or negative (trajectory in the right subplot). As a consequence, it is no longer possible to describe the solution trajectories of the middle or right subplot by a continuous deformation (homeomorphism) of solution trajectories of the left subplot. Therefore, the system shown in (6.8) is not structurally stable.

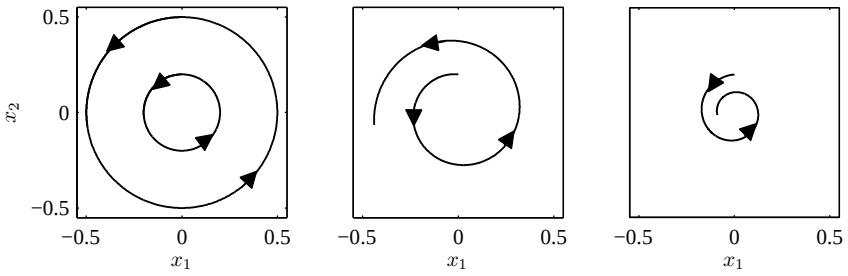


Figure 6.2: Trajectories of three dynamical systems.

In relation to numerical simulation of systems that are not structurally stable, even the smallest rounding error in the representation of the system may significantly alter its behavior.

Two important restrictions of Morse-Smale systems are exploited in this paper. First, the vector field has a finite number of singular points, each hyperbolic. A *hyperbolic singular point* is a singular point such that in local coordinates the matrix of partial derivatives of the vector field has eigenvalues with nonzero real parts (the eigenvalues of (6.8) are not hyperbolic). This property allows the system to be split up into a stable and an unstable subsystem, which can be analyzed separately. Second, the stable and unstable manifolds associated with a singular point have transversal intersection. This restriction is necessary to obtain a structurally stable vector field.

Remark 9*: To the best of authors' knowledge, there exists no method to check if a system is Morse-Smale. However, if the system is second order and one finds a Lyapunov function, it is Morse-Smale. For a system of dimension greater than two, if there is a Lyapunov function then it can be approximated arbitrary closely by a Morse-Smale vector field [16]. For linear systems, it is necessary and sufficient to check if all eigenvalues are hyperbolic.*

To summarize, it is chosen to only consider Morse-Smale systems in this paper, as they are structurally stable, and there exists an open set of Lyapunov functions for all Morse-Smale systems, which is important for the subdivisioning.

In the following, we reason about the size of the class of Morse-Smale systems (or vector fields).

Morse-Smale systems are dense in systems of less than or equal to two dimensions according to the following two theorems.

Theorem 8 ([20])*: In order that the vector field ξ be structurally stable on the compact 2-dimensional manifold M it is necessary and sufficient that the vector field is Morse-Smale.*

Theorem 9 ([20])*: The set of all structurally stable systems is open and dense in the space of all systems defined on a 2-dimensional manifold M.*

From Theorem 8 we conclude that Morse-Smale systems are robust in the sense that its behavior does not change dramatically after being perturbed slightly, as it is structurally stable (explained next). This also implies that the subdivisioning will be insensitive to small perturbations. This is a desirable property, as small perturbations always exist for physical systems. From Theorem 9 we conclude that for two dimensions all dynamical systems can be approximated by the Morse-Smale system with arbitrary accuracy.

We are of course interested in studying systems having dimension greater than two; however, note that we want to verify control systems. Control systems are almost always designed using some Lyapunov-based method, i.e., there exists a Lyapunov function for the considered system, which implies that it is Morse-Smale. Therefore, we conclude that it is reasonable to only study Morse-Smale systems in this analysis.

Timed Automata

We abstract dynamical systems by timed automata. Therefore, a definition of a timed automaton is provided in the following Alur et al. [21]. Before defining it, a set of diagonal-free clock constraints $\Psi(C)$ for the set C of clocks is defined. $\Psi(C)$ contains all invariants and guards of the timed automaton; consequently, it is described by the

following grammar

$$\psi ::= c \bowtie k | \psi_1 \wedge \psi_2, \tag{6.9a}$$

where

$$c \in C,\ k \in \mathbb{R}_{\geq 0},\ \text{and}\ \bowtie \in \{\leq, <, =, >, \geq\}. \tag{6.9b}$$

Note that the clock constraint k should usually be a rational number, but in this paper, no effort is done to convert the clock constraints into rational numbers. However, any real number can be approximated by a rational number with an arbitrary small error $\epsilon > 0$.

Definition 46 (Timed Automaton)**.** A timed automaton $\mathcal{A}$ is a tuple $(E, E_0, C, \Sigma, I, \Delta)$, where

- E is a finite set of locations, and $E_0 \subseteq E$ is the set of initial locations.
- C is a finite set of clocks.
- Σ is the input alphabet.
- $I : E \to \Psi(C)$ assigns invariants to locations, where $\Psi(C)$ is the set of all clock constraints in (6.9).
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations. A transition relation is a tuple $(e, G_{e\to e'}, \sigma, R_{e\to e'}, e')$ that assigns an edge between two locations, where e is the source location, e' is the destination location, $G_{e\to e'} \in \Psi(C)$ is the guard set, σ is a symbol in the alphabet Σ, and $R_{e\to e'} \subseteq C$ is a subset of clocks.

The semantics of a timed automaton is defined in the following, adopting the notion of [22].

Definition 47 (Clock Valuation)**.** A clock valuation on a set of clocks C is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation v_0 is given by $v_0(c) = 0$ for all $c \in C$. For a valuation v, $d \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + d$ and $v[R]$ are defined as

$$(v + d)(c) = v(c) + d, \tag{6.10a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{6.10b}$$

Definition 48 (Semantics of Clock Constraint)**.** A clock constraint in $\Psi(C)$ is a set of clock valuations $\{v : C \to \mathbb{R}_{\geq 0}\}$ given by

$$[\![c \bowtie k]\!] = \{v : C \to \mathbb{R}_{\geq 0} | v(c) \bowtie k\} \tag{6.11a}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]. \tag{6.11b}$$

For convenience we denote $v \in [\![\psi]\!]$ by $v \models \psi$.

Definition 49 (Semantics of Timed Automaton)**.** The semantics of a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ is the transition system $[\![\mathcal{A}]\!] = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_\mathrm{s} \cup T_\mathrm{d})$ given by

$$
\begin{aligned}
S &= \{(e, v) \in E \times \mathbb{R}^C_{\geq 0} |\ v \models I(e)\} \\
S_0 &= \{(e, v) \in E_0 \times \mathbb{R}^C_{\geq 0} |\ v = v_0\} \\
T_\mathrm{s} &= \{(e, v) \xrightarrow{\sigma} (e', v') |\ \exists (e, G_{e\to e'}, \sigma, R_{e\to e'}, e') \in \Delta : v \models G_{e\to e'},\, v' = v[R_{e\to e'}]\} \\
T_\mathrm{d} &= \{(e, v) \xrightarrow{d} (e, v + d) |\ \forall d' \in [0, d] : v + d' \models I(e)\}.
\end{aligned}
$$

Analog to the solution of (6.1), shown in (6.2), is a run of a timed automaton, which is defined in the following.

Definition 50 (Run of Timed Automaton)**.** A run of a timed automaton $\mathcal{A}$ is a possibly infinite sequence of alternations between time steps and discrete steps on the following form

$$(e_0, v_0) \xrightarrow{d_1} (e_0, v_1) \xrightarrow{\sigma_1} (e_1, v_2) \xrightarrow{d_2} \dots \tag{6.12}$$

where $d_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$. The multifunction describing runs of a timed automaton $\phi_\mathcal{A} : \mathbb{R}_{\geq 0} \times E_0 \to 2^E$, defined by $e \in \phi_\mathcal{A}(t, e_0)$ if and only if there exists a path in $[\![\mathcal{A}]\!]$ initialized in (e_0, v_0) that reaches the location e at time $t = \sum_i d_i$.

We use the notation of the run of a timed automaton, to define its reachable locations.

Definition 51 (Reachable Set of Timed Automaton)**.** The reachable set of a timed automaton $\mathcal{A}$ with initial locations E_0 on the time interval $[t_1, t_2]$ is defined by

$$\mathrm{Reach}_{[t_1,t_2]}(\mathcal{A}, E_0) = \{e \in E |\ \exists t \in [t_1, t_2], \exists e_0 \in E_0, \text{ such that } e \in \phi_\mathcal{A}(t, e_0)\}. \tag{6.13}$$

This concludes the preliminary definitions. The next section explains how the state space of a dynamical system should be subdivided.

3 Generation of Finite Subdivision

The main idea of this work is to design an abstraction procedure, which exploits the knowledge of the flow of the dynamical system. Therefore, it is proposed to subdivide the state space into a finite number of cells by intersecting slices defined as the set-difference of positive and negative invariant sets. This ensures a unidirectional flow through the boundaries of the cells, see Definition 44. Another approach to subdividing the state space of a hybrid system by a family of functions is developed in [23]. In this work, the state space is subdivided by a family of immersed submanifolds, so called leafs of the foliation. Each leave is of codimension 1, and it is transversal to the studied vector field. Such a foliation can be generated as an inverse image of a regular value under a function. Consequently, a collection of foliations defines a subdivision. A regular point of a map f is a point where the differential of f is surjective. A point v, in the image of f, is a regular value if its pre-image contains regular points only (i.e. no critical points), see Definition 74.

In most previously proposed methods for abstracting dynamical systems, the subdivision of the state space is done without considering the dynamics of the system. In contrast to this, the proposed method is mainly concerned with the subdivision of the state space according to the system dynamics.

Definition 52 (Slice)**.** A nonempty set S is a slice if there exist two open sets A_1 and A_2 such that

1. A_1 and A_2 are positively or negatively invariant,
2. A_1 is a proper subset of A_2, and
3. $S = \mathrm{cl}(A_2 \backslash A_1)$.

The slices are defined to be set-differences of positive or negative invariant sets to ensure that the vector field is transversal, see Definition 55, to the boundaries of the slices. Figure 6.3 illustrates a slice and the two positive invariant sets, A_1 and A_2, which generate it. This construction of slices ensures a nonzero minimum time for staying in each slice unless it contains an equilibrium point. We know from the definition of the Lyapunov function that there are no limit cycles in the set $A_2 \backslash A_1$. If there was a limit cycle and a and b were two points on it, then the system would first reach a, then b and a again. However, as the Lyapunov function is strictly decreasing it would imply that $\varphi(a) > \varphi(b) > \varphi(a)$, which cannot be true.

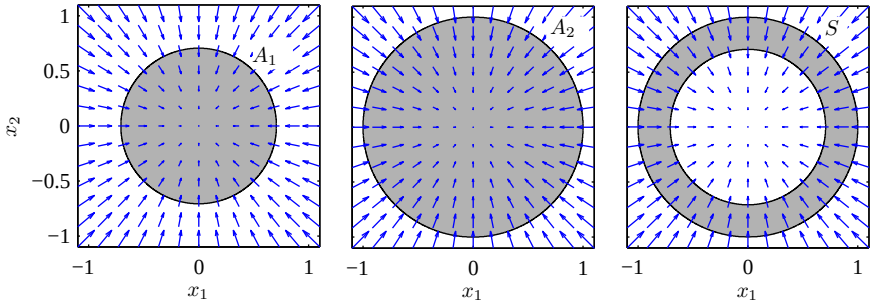


Figure 6.3: Illustration of a phase plot of a dynamical system (blue arrows), two positive invariant sets, A_1 and A_2 (gray disks), and a slice $S = \mathrm{cl}(A_2 \backslash A_1)$ to the right.

To devise a subdivision of a state space, we need to define finite collections of slices. These collections are called slice-families.

Definition 53 (Slice-Family)**.** A slice-family $\mathcal{S}$ is a finite collection of slices generated by the positive or negative invariant open sets $A_1 \subset A_2 \subset \cdots \subset A_k$ covering the entire state space of the system $\Gamma = (X, f)$. Thereby, $S_1 = \mathrm{cl}(A_1)$, $S_2 = \mathrm{cl}(A_2 \backslash A_1), \ldots, S_k = \mathrm{cl}(A_k \backslash A_{k-1})$, and $X \subseteq A_k$.
For convenience $|\mathcal{S}|$ is defined to be the number of slices in the slice-family $\mathcal{S}$.

We say that the slice-family $\mathcal{S}$ is generated by the sets $\{A_i | i \in \boldsymbol{k}\}$. We address the existence and generation of these sets in Section 7.

A function is associated to each slice-family $\mathcal{S}$ to provide an easy way of describing the boundary of a slice. Such a function is called a subdivisioning function.

Definition 54 (Subdivisioning Function)**.** Let $\mathcal{S}$ be a slice-family, then a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \mathrm{Cr}(f)$ is a subdivisioning function associated to $\mathcal{S}$ if for any positive or negative invariant set A_i generating $\mathcal{S}$ there exist $a_i, a_i' \in \mathbb{R} \cup \{-\infty, \infty\}$ such that

$$\varphi^{-1}([a_i, a_i']) = \mathrm{cl}(A_i) \tag{6.14}$$

and a_i, a_i' are regular values of φ, see Definition 74. By regular level set theorem, the boundary $\varphi^{-1}(a_i)$ of A_i is an embedded smooth submanifold of $\mathbb{R}^n$ [24]. We index the regular values such that $a_i < a_j$ if and only if $i < j$.

The proposed method heavily relies on transversal intersections; therefore, a formal definition of transversal intersection is provided in the following. After the definition, it is geometrically interpreted to clarify.

Definition 55 (Transversal Intersection [19])**.** Suppose that N_1 and N_2 are embedded submanifolds of M. We say that N_1 intersects N_2 transversally if, whenever $p \in N_1 \cap N_2$, we have $T_p(N_1) + T_p(N_2) = T_p(M)$. (The sum is not direct, just the set of sums of vectors, one from each of the two subspaces of the tangent space $T_p(M)$.)

In the left subplot of Figure 6.4, level sets of two subdivisioning functions (hence two embedded submanifolds of $\mathbb{R}^2$) are illustrated. They intersect in the point p; however, their tangents (black lines) are identical. This implies that their tangent vectors only span one dimension at p, i.e., $T_p(N_1) + T_p(N_2) \neq T_p(M)$. Therefore, this intersection is not transversal. Note that with an arbitrary small perturbation, the intersection of the two level sets will be empty, as shown in the middle subplot (This perturbation is given by a smooth map, see Theorem 2.1 in [19]).

In the right subplot Figure 6.4, two level sets intersecting at point p are illustrated. Their tangent vectors (black lines) span $\mathbb{R}^2$, i.e., the level sets intersect transversally. Note that two manifolds that do not intersect are also transversal.

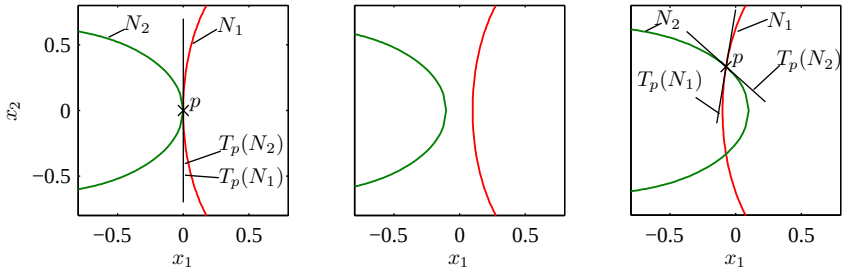


Figure 6.4: Illustration of two manifolds N_1 (red) and N_2 (green). In the left subplot N_1 and N_2 intersect; however, not transversally. In the middle subplot N_1 and N_2 do not intersect, but are transversal. In the right subplot N_1 and N_2 intersect transversally.

It is desired to obtain cells from the slices, and this is done by intersecting slices.

Definition 56 (Transversal Intersection of Slices)**.** We say that the slices S_1 and S_2 intersect each other transversally and write

$$S_1 \pitchfork S_2 = S_1 \cap S_2 \tag{6.15}$$

if their boundaries, $\mathrm{bd}(S_1)$ and $\mathrm{bd}(S_2)$, intersect each other transversally.

Remark 10*: A subdivision is robust, if any two slices intersect each other transversally, since they still intersect each other transversally after an arbitrary small perturbation. Hence, robustness is the reason for considering only transversal intersections in the proposed subdivision.*

Definition 57 (Extended Cell)**.** Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of k slice-families and let $\mathcal{G}(\mathcal{S}) \equiv \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\}$.
Denote the j^{th} slice in $\mathcal{S}^i$ by S^i_j and let $g \in \mathcal{G}(\mathcal{S})$. Then

$$e_{\text{ex},g} = \pitchfork_{i=1}^{k} S^i_{g_i} \tag{6.16}$$

where g_i is the i^{th} element of the vector g. Any nonempty set $e_{\text{ex},g}$ is called an extended cell.

The set $\mathcal{G}(\mathcal{S})$, defined above, is used in the remainder of the paper.

The cells in (6.16) are denoted by extended cells, since the transversal intersection of slices may form multiple disjoint sets in the state space. This is illustrated in Figure 6.5, where a two-dimensional state space is subdivided utilizing two slice-families (red and green).

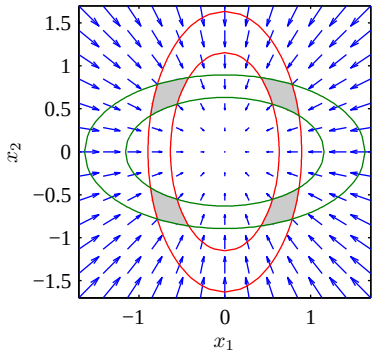


Figure 6.5: Phase plot (blue arrows) of a two-dimensional system and a subdivision generated utilizing two slice-families (red and green lines). The gray shaded area illustrates one extended cell, which consists of 4 connected components.

The next example clarifies the indexing of the extended cells shown in Definition 57.

Example 6 (Indexing Extended Cells)**.** Given three slice-families $\{\mathcal{S}^i | i \in \{1, 2, 3\}\}$, an extended cell is indexed according to the ordering of the slices defining it, as shown below

$$e_{\text{ex},(9,5,27)} = S^1_9 \pitchfork S^2_5 \pitchfork S^3_{27}. \tag{6.17}$$

The vector g from Definition 57 equals $(9, 5, 27)$ in this example, since $e_{\text{ex},(9,5,27)}$ is generated from slice number 9 in $\mathcal{S}^1$, slice number 5 in $\mathcal{S}^2$, and slice number 27 in $\mathcal{S}^3$.

It is desired to have cells, which are connected. Therefore, the following is defined.

Definition 58 (Cell)**.** A cell is a connected component of an extended cell

$$\bigcup_h e_{(g,h)} = e_{\text{ex},g}, \tag{6.18a}$$

where

$$e_{(g,h)} \cap e_{(g,h')} = \emptyset \qquad \forall h \neq h'. \tag{6.18b}$$

We say that the slices $S^1_{g_1}, \ldots, S^k_{g_k}$ generate the cell $e_{(g,h)}$.

Figure 6.6 illustrates a two-dimensional state space subdivided by two slice-families and provides a geometric interpretation of the definitions related to the subdivisioning of a state space. The symbols of cells and level-sets of subdivisioning functions from the previous definitions are added to clarify their meaning. Note that subscript of a^i_j refers to the number in the slice families, whereas the superscript is the number of slice family, i.e., a^i_j is the j^{th} regular value associated with subdivisioning function φ^i.

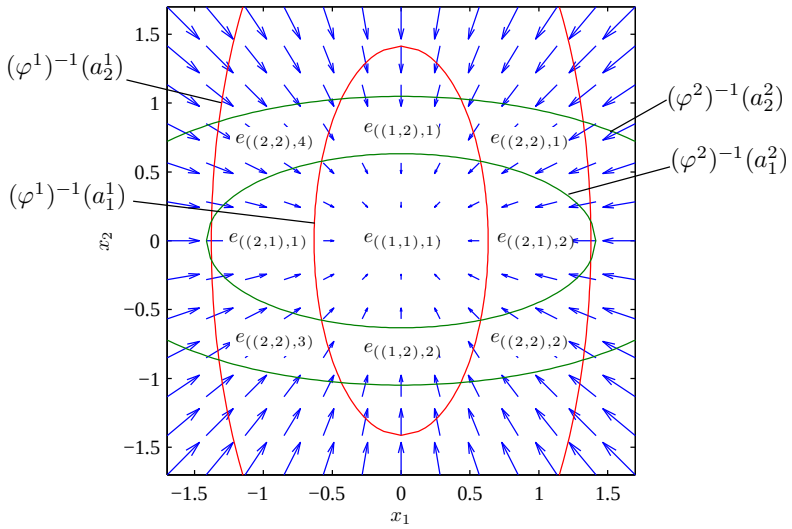


Figure 6.6: Phase plot (blue arrows) of a two-dimensional state space. The state space is subdivided utilizing two slice-families, generated by the subdivisioning functions φ^1 (red) and φ^2 (green). Each cell of the subdivision is numbered according to Definition 57 and Definition 58.

A finite subdivision based on the transversal intersection of slices is defined in the following.

Definition 59 (Finite Subdivision)**.** Let $\mathcal{S}$ be a collection of slice-families, $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. We define a finite subdivision $E(\mathcal{S})$ by

$$e \in E(\mathcal{S}) \tag{6.19}$$

if and only if e is a connected component of an extended cell.

Based on the definitions provided in this section, a procedure for obtaining a timed automaton from a finite subdivision of a state space is presented in the next section.

4 Generation of Timed Automaton from Finite Subdivision

The purpose of this section is to explain how a timed automaton $\mathcal{A}$ is generated from a finite subdivision $E(\mathcal{S})$ of the state space of a system Γ. For this, we use the abstraction procedure presented in [6]; nevertheless, we exclude the clock and constraints related to the time of traversing a cell. However, these can be added to improve accuracy. The reason why these clocks are removed is that they destroy the compositional structure of the timed automaton and requires computation of more guards and invariants.

First, we define an abstraction function associating each cell of the subdivision $E(\mathcal{S})$ to a location of a timed automaton.

Definition 60 (Abstraction Function)**.** Let $E(\mathcal{S}) = \{e_i | i \in \boldsymbol{m}\}$ for some $m \in \mathbb{N}$ be a finite subdivision of the state space $X \in \mathbb{R}^n$ and $E(\mathcal{S}) = \{e_i | i \in \boldsymbol{m}\}$ be the locations of a timed automaton. Then an abstraction function for $(X, E(\mathcal{S}))$ is a multifunction $\alpha_E : X \to 2^{E(\mathcal{S})}$ defined by

$$\alpha_E(x) = \{e_i \in E(\mathcal{S}) \mid x \in e_i\}. \tag{6.20}$$

We use this abstraction function and generate a timed automaton according to the following procedure.

Procedure 1 (Generation of a Timed Automaton)*: Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a finite collection of slice-families. Then the timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ generated by the subdivision $E(\mathcal{S})$ is defined by*

- ***Locations:*** *The locations of $\mathcal{A}$ are given by*

$$E = E(\mathcal{S}). \tag{6.21}$$

  This means that a location $e_{(g,h)}$ is associated with the cell $e_{(g,h)} = \alpha_K^{-1}(e_{(g,h)})$ of the subdivision $E(\mathcal{S})$, see Definition 60.

- ***Clocks:*** *The number of clocks equals the number of slice-families, i.e., $C = \{c^i | i \in \boldsymbol{k}\}$.*

- ***Invariants:*** *In each location $e_{(g,h)}$, there are up to k invariants. We impose an invariant, whenever there is an upper bound for the time of staying in a slice generating the cell $e_{(g,h)}$*

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \bar{t}_{S^i_{g_i}}, \tag{6.22}$$

  where $\bar{t}_{S^i_{g_i}} \in \mathbb{R}_{\geq 0}$ is an upper bound on the time for staying in $S^i_{g_i}$.

- ***Input Alphabet:*** *The input alphabet Σ consists of k symbols $\{\sigma^i | i \in \boldsymbol{k}\}$. Note that σ_i is associated with transitions between two slices in the slice-family $\mathcal{S}^i$.*

- ***Transition relations:*** *If a pair of locations $e_{(g,h)}$ and $e_{(g',h')}$, associated with the cells $e_{(g,h)}$ and $e_{(g',h')}$, satisfy the following two conditions*

  1. *$e_{(g,h)}$ and $e_{(g',h')}$ are adjacent, that is $e_{(g,h)} \cap e_{(g',h')} \neq \emptyset$, and*
  2. *$g'_i \leq g_i$ for all $i \in \boldsymbol{k}$.*

  Then there is a transition relation

$$\delta_{(g,h)\to(g',h')} = (e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, e_{(g',h')}), \tag{6.23a}$$

where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{6.23b}$$

and $\underline{t}_{S^i_{g_i}} \in \mathbb{R}_{\geq 0}$ *is a lower bound on the time for staying in* $S^i_{g_i}$*. Note that* $g_i - g'_i = 1$ *whenever a transition labeled* σ^i *is taken. If* $g_i - g'_i = 0$*, we stay in the slice from the* i^{th} *slice-family generating cell* $e_{(g,h)}$*; hence, no active guards are on* c^i*.*
Note that there is only constraints on the i *clock if* $g_i - g'_i = 1$*, since otherwise we do not exit a slice from the* i^{th} *slice-family.*
Let $i \in \boldsymbol{k}$*. We define* $R_{(g,h)\to(g',h')}$ *by*

$$c^i \in R_{(g,h)\to(g',h')} \tag{6.23c}$$

iff $g_i - g'_i = 1$*.*

The calculation of $\underline{t}_{S^i_{g_i}}$ *and* $\overline{t}_{S^i_{g_i}}$ *can be accomplished in multiple ways, yielding different properties of the abstraction. This is explained in details in Section 6, where conditions for generating sound and complete abstractions are provided.*

For convenience the following notion is introduced.

Definition 61. Let $\mathcal{S}$ be a finite collection of slice-families $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then $\mathcal{A}(\mathcal{S})$ is the timed automaton generated by $\mathcal{S}$ according to Procedure 1.

Definition 62. A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ has locations given by

$$E = E_{\text{ex}}(\mathcal{S}), \tag{6.24}$$

where a location $e_{\text{ex},g} \in E_{\text{ex}}(\mathcal{S})$ is associated with the extended cell $e_{\text{ex},g}$ generated by the slice-family $\mathcal{S}$; hence, $e_{\text{ex},g} = \alpha^{-1}_{K_{\text{ex}}}(e_{\text{ex},g})$.

5 Properties of the Generated Timed Automaton

A timed automaton generated from the presented procedure possesses some salient properties, due to the way state spaces are subdivided. Some of these properties are presented in this section. The section consists of four subsections, each explaining one of the properties.

Determinism of Abstractions

The considered systems have unique maximal solutions, see Definition 41. Therefore, we also determine when a subdivision generates a deterministic timed automaton.

Proposition 18 (Deterministic Timed Automaton)*: The timed automaton* $\mathcal{A}(\mathcal{S})$ *is deterministic, if and only if for each cell* $e_{(g,h)} \in K(\mathcal{S})$ *and for all* $i \in \boldsymbol{k}$ *the set*

$$e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i - 1}) \tag{6.25}$$

is connected.

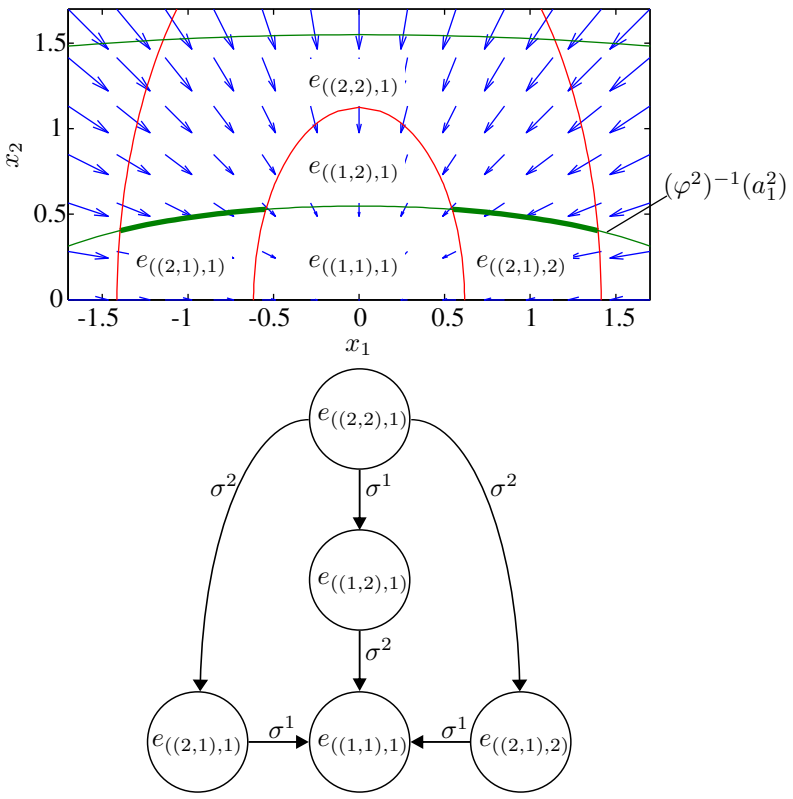


Figure 6.7: Subdivision of a two-dimensional state space using two slice-families $\mathcal{S}^1$ (red) and $\mathcal{S}^2$ (green), and an illustration of the resulting nondeterministic timed automaton (for simplicity only the names of the locations and symbols on the transitions are shown in the figure). The set $e_{((2,2),1)} \bigcap (\varphi^2)^{-1}(a_1^2)$ is illustrated with a bold green line, to emphasize that Proposition 18 is not satisfied, as the set (6.25) is not connected.

We show how to check if an abstraction is deterministic in Proposition 20; however, first we present an example to clarify Proposition 18.

Example 7. Consider the subdivision of a two-dimensional state space shown in Figure 6.7 generated using two slice-families $\mathcal{S}^1$ (red) and $\mathcal{S}^2$ (green). A timed automaton generated from the subdivision is shown in the bottom of the figure, excluding time information.

From the figure it is seen that the symbol σ^2 determines two transitions in the timed automaton from $e_{((2,2),1)}$: A transition to $e_{((2,1),1)}$ and a transition to $e_{((2,1),2)}$. The reason for this is that two cells are reachable from $e_{((2,2),1)}$ by crossing $(\varphi^2)^{-1}(a_1^2)$, as $e_{((2,2),1)} \bigcap (\varphi^2)^{-1}(a_1^2)$ (bold green line) consists of two connected components. This implies that Proposition 18 is not satisfied; hence, a timed automaton generated by this subdivision is nondeterministic.

For linear systems, it is possible to check if Proposition 18 is satisfied when quadratic

Lyapunov functions are used as subdivisioning functions. This is explained in the following, using a proposition from [25] that determine when two quadratic forms intersect.

Proposition 19 ([25]): *Suppose* $P^1 = (P^1)^T > 0$, $P^2 = (P^2)^{\mathrm{T}} > 0$, *and let* $\varphi^1(x) = x^{\mathrm{T}} P^1 x$ *and* $\varphi^2(x) = x^{\mathrm{T}} P^2 x$. *Define* $\overline{\gamma}$ *to be the solution to the following optimization problem*

$$\text{minimize } \overline{\gamma} \tag{6.26a}$$

$$\text{subject to } P^2 - \overline{\gamma} P^1 \leq 0, \qquad \overline{\gamma} > 0 \tag{6.26b}$$

and define $\underline{\gamma}$ *to be the solution to the following optimization problem*

$$\text{maximize } \underline{\gamma} \tag{6.27a}$$

$$\text{subject to } P^1 \underline{\gamma} - P^2 \leq 0, \qquad \underline{\gamma} > 0. \tag{6.27b}$$

Then the level set $(\varphi^2)^{-1}(a_2)$ *generated for a regular value* a_2 *intersects* $(\varphi^1)^{-1}(a_1)$ *if and only if* $a_2 \in [a_1\underline{\gamma}, a_1\overline{\gamma}]$, *and* $(\varphi^2)^{-1}(a_2)$ *intersects* $(\varphi^1)^{-1}(a_1)$ *transversally if and only if* $a_2 \in (a_1\underline{\gamma}, a_1\overline{\gamma})$.

This is illustrated in Figure 6.8, where the level sets $(\varphi^1)^{-1}(a_1)$, $(\varphi^2)^{-1}(a_1\underline{\gamma})$, and $(\varphi^2)^{-1}(a_1\overline{\gamma})$ are drawn.

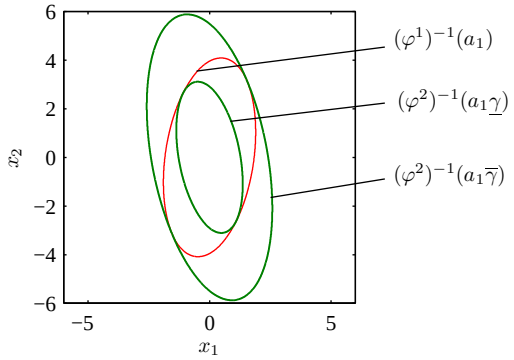


Figure 6.8: Illustration of a level set $(\varphi^1)^{-1}(a_1)$ (red) and the level sets $(\varphi^2)^{-1}(a_1\underline{\gamma})$, and $(\varphi^2)^{-1}(a_1\overline{\gamma})$ (green) intersecting $(\varphi^1)^{-1}(a_1)$.

From Figure 6.7 it is seen that the timed automaton becomes nondeterministic, when a level set only intersects some, but not all level sets from the other slice families. Using this fact, the following proposition follows directly from Proposition 19, and is left without proof.

Proposition 20: *Let* $\mathcal{A}(\mathcal{S})$ *be an abstraction of a linear system* $\Gamma = (X, f)$. *Let* $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ *be a collection of slice-families. Associate to each slice-family* $\mathcal{S}^i$ *a quadratic subdivisioning function* $\varphi^i(x) = x^{\mathrm{T}} P^i x$, *and let* $\mathcal{S}^i$ *be generated using the regular values* $\{a^i_k | k \in \{1, \ldots, |\mathcal{S}^i|\}\}$. *For all* $i, j \in \boldsymbol{k}$, *define* $\overline{\gamma}^{ij}$ *to be the solution to the following optimization problem*

$$\text{minimize } \overline{\gamma}^{ij} \tag{6.28a}$$

$$\text{subject to } P^j - \overline{\gamma}^{ij} P^i \leq 0, \qquad \overline{\gamma}^{ij} > 0 \tag{6.28b}$$

and define $\underline{\gamma}^{ij}$ *to be the solution to the following optimization problem*

$$\text{maximize } \underline{\gamma}^{ij} \tag{6.29a}$$

$$\text{subject to } P^i\underline{\gamma}^{ij} - P^j \leq 0, \qquad \underline{\gamma}^{ij} > 0. \tag{6.29b}$$

Then $\mathcal{A}(\mathcal{S})$ *is deterministic if and only if for all* $i \neq j \in \boldsymbol{k}$ *and for all* $a^j \in \{a^j_k | k \in \{1, \ldots, |\mathcal{S}^j|\}\}$

$$a^j \in (a^i\underline{\gamma}^{ij}, a^i\overline{\gamma}^{ij}) \qquad \forall a^i \in \{a^i_k | k \in \{1, \ldots, |\mathcal{S}^i|\}\} \text{ or} \tag{6.30a}$$

$$a^j \notin (a^i\underline{\gamma}^{ij}, a^i\overline{\gamma}^{ij}) \qquad \forall a^i \in \{a^i_k | k \in \{1, \ldots, |\mathcal{S}^i|\}\}. \tag{6.30b}$$

Remark that the optimization problems used in Proposition 20 can be solved using standard tools for solving linear optimization problems.

Compositionality of Abstractions

Under certain conditions it is possible to generate the timed automaton as a parallel composition of multiple timed automata.

Definition 63 (Parallel Composition of Timed Automata)**.** The parallel composition of two timed automata, $\mathcal{A}_i = (E_i, E_{0,i}, C_i, \Sigma_i, I_i, \Delta_i)$ for $i = 1, 2$ with transition relations $(e_i, G_{i,e_i \to e_i'}, \sigma^i, R_{i,e_i \to e_i'}, e_i')$, is denoted $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$ and is a timed automaton $(E, E_0, C, \Sigma, I, \Delta)$, where:

- $E = E_1 \times E_2$.
- $E_0 = E_{0,1} \times E_{0,2}$.
- $C = C_1 \cup C_2$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$.
- $I : E \to \Psi(C)$, where $I(e_1, e_2) = I_1(e_1) \wedge I_2(e_2)$.
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations, where $(e, G_{e \to e'}, \sigma, R_{e \to e'}, e')$ is defined by the following
  1. If $\sigma \in \Sigma_1 \cap \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{1,e_1 \to e_1'} \wedge G_{2,e_2 \to e_2'}$, $R_{e \to e'} = R_{1,e_1 \to e_1'} \cup R_{2,e_2 \to e_2'}$, and $e' = (e_1', e_2')$.
  2. If $\sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{1,e_1 \to e_1'}$, $R_{e \to e'} = R_{1,e_1 \to e_1'}$, and $e' = (e_1', e_2)$.
  3. If $\sigma \notin \Sigma_1$ and $\sigma \in \Sigma_2$ then $e = (e_1, e_2)$, $G_{e \to e'} = G_{2,e_2 \to e_2'}$, $R_{e \to e'} = R_{2,e_2 \to e_2'}$, and $e' = (e_1, e_2')$.

Proposition 21*: Let* $\mathcal{A}_{\text{ex}}(\mathcal{S})$ *be a timed automaton and let the slices of* $\mathcal{S}$ *be generated such that for each pair* $(S^i_{g_i}, S^j_{g_j})$*, with* $i, j \in \boldsymbol{k}$*,* $g_i \in \{1, \ldots, |\mathcal{S}^i|\}$*,* $g_j \in \{1, \ldots, |\mathcal{S}^j|\}$*, we have*

$$S^i_{g_i} \pitchfork S^j_{g_j} \neq \emptyset \qquad \forall i \neq j. \tag{6.31}$$

Then, $\mathcal{A}_{\text{ex}}(\mathcal{S})$ *is isomorphic to the parallel composition of* k *timed automata each generated by one slice-family* $\mathcal{S}^i$ *having an alphabet* $\Sigma_i = \{\sigma^i\}$*.*

Note that case 1 in Definition 63, where $\sigma \in \Sigma_i \cap \Sigma_j$ is never satisfied in this work, as $\Sigma_i \cap \Sigma_j = \emptyset$ for all $i \neq j$.

Remark 11*: A parallel composition of timed automata $\mathcal{A}_i(\mathcal{S}^i)$ for $i \in \boldsymbol{k}$ is similar to the intersection of slices in the slice-families $\mathcal{S}^i$. Therefore, the intersection of slices should be nonempty to let the locations of the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ be such a parallel composition, as stated in Proposition 21.*

The satisfaction of Proposition 21 can be checked using the following proposition.

Proposition 22*: Let $\mathcal{A}(\mathcal{S})$ be an abstraction of a linear system $\Gamma = (X, f)$. Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families. Associate to each slice-family $\mathcal{S}^i$ a quadratic subdivisioning function $\varphi^i(x) = x^{\mathrm{T}} P^i x$, and let $\mathcal{S}^i$ be generated using the regular values $\{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}$. For all $i, j \in \boldsymbol{k}$, define $\overline{\gamma}^{ij}$ to be the solution to the following optimization problem*

$$\text{minimize } \overline{\gamma}^{ij} \tag{6.32a}$$

$$\text{subject to } P^j - \overline{\gamma}^{ij} P^i \leq 0, \qquad \overline{\gamma}^{ij} > 0 \tag{6.32b}$$

and define $\underline{\gamma}^{ij}$ to be the solution to the following optimization problem

$$\text{maximize } \underline{\gamma}^{ij} \tag{6.33a}$$

$$\text{subject to } P^i \underline{\gamma}^{ij} - P^j \leq 0, \qquad \underline{\gamma}^{ij} > 0. \tag{6.33b}$$

Then Proposition 21 is satisfied if and only if for all $i \neq j \in \boldsymbol{k}$ and for all $a^j \in \{a_k^j | k \in \{1, \ldots, |\mathcal{S}^j|\}\}$

$$a^j \in (a^i \underline{\gamma}^{ij}, a^i \overline{\gamma}^{ij}) \qquad \forall a^i \in \{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}. \tag{6.34a}$$

The property that $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is isomorphic to the parallel composition of k timed automata is very important for computations, since it allows parallel verification of the k timed automata each with only one clock. Furthermore, it makes it possible to sequentially add slice-families to the abstraction, to replace, and to refine slice-families to improve the accuracy of the abstraction.

The parallel composition of timed automata also allows the sequential verification of the abstraction. We show this in terms of safety in the following.

Definition 64 (Safety)**.** Given a timed automaton $\mathcal{A}(\mathcal{S})$ and a set of unsafe locations E_{US}. The timed automaton $\mathcal{A}(\mathcal{S})$ is said to be safe if

$$\text{Reach}_{[0,\infty)}(\mathcal{A}(\mathcal{S}), E_0) \cap E_{\text{US}} = \emptyset. \tag{6.35}$$

Proposition 23*: Let $\mathcal{A}_{\text{ex}}(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1) || \ldots || \mathcal{A}_k(\mathcal{S}^k)$ be a timed automaton and let the timed automaton $\mathcal{A}_1(\mathcal{S}^1) || \ldots || \mathcal{A}_j(\mathcal{S}^j)$ be safe, for some $j \in \boldsymbol{k}$. Then, $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is also safe.*

This proposition is quite intuitive, when considering how the timed automaton is generated from positive invariant sets as shown in the following example.

Example 8. Consider a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by two slice-families abstracting a two-dimensional system as shown in Figure 6.9 (left subplot). Its initial states are illustrated as gray areas and the unsafe states are illustrated as black areas.

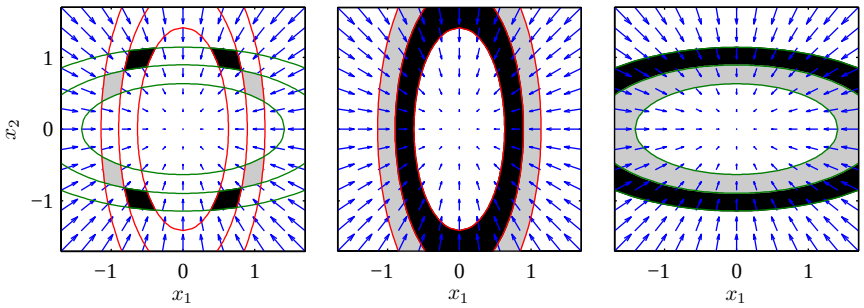


Figure 6.9: Illustration of subdivision associated with $\mathcal{A}_{\text{ex}}(\mathcal{S})$ (left subplot), $\mathcal{A}(\mathcal{S}^1)$ (middle subplot), and $\mathcal{A}(\mathcal{S}^2)$ (right subplot). The gray areas illustrate initial states and the black are illustrates unsafe states.

We compose $\mathcal{A}_{\text{ex}}(\mathcal{S})$ into $\mathcal{A}(\mathcal{S}^1)$ (middle subplot) and $\mathcal{A}(\mathcal{S}^2)$ (right subplot). We observe that $\mathcal{A}(\mathcal{S}^2)$ is safe; hence, $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is also safe, which is seen in the left subplot.

Bisimilarity of Abstractions Generated from Cells and Extended Cells

Under certain conditions, the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is bisimilar to the timed automaton $\mathcal{A}(\mathcal{S})$, bisimilarity is defined in Definition 75. In the next proposition we say that the timed automata $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are related by bisimulation.

Proposition 24: *Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families, and φ^i be a subdivisioning function for $\mathcal{S}^i$. A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by extended cells is bisimilar to a timed automaton $\mathcal{A}(\mathcal{S})$ generated by cells if for each cell $e_{(g,h)}$ and each $i \in \boldsymbol{k}$*

$$e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i-1}) \neq \emptyset \qquad \forall\, h \text{ or} \tag{6.36a}$$

$$e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i-1}) = \emptyset \qquad \forall\, h. \tag{6.36b}$$

If (6.36) holds, then all cells in each extended cell have the same symbols on their outgoing transitions; hence, $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are bisimilar. The following example clarifies this proposition.

Example 9. To illustrate the use of Proposition 24 three different subdivisions of a two-dimensional state space are shown in Figure 6.10.

In the subdivisions shown in the left and right side of Figure 6.10, the conditions shown in (6.36) are satisfied for all g and h. In the subdivision shown in the middle side of Figure 6.10, the constraints shown in (6.36) are not satisfied for e.g. $g = (2, 2, 2)$ (shaded region), as $(\varphi^3)^{-1}(a^3_1)$ (inner black line) does not intersect all cells in $e_{\text{ex},(2,2,2)}$.

Convergence Check via Partial Order

We can define a partial order for the locations of a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by Procedure 1 as follows. Recall that a partial order on a set X, denoted $\sqsubseteq$, is a relation on X that is reflexive, transitive, and anti-symmetric.

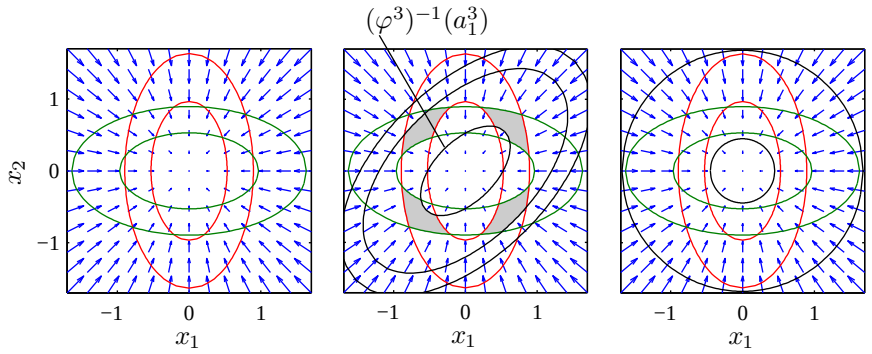


Figure 6.10: Illustration of three different subdivisions of a two-dimensional state space. The left and right subdivisions satisfy Proposition 24. The middle subdivision does not satisfy Proposition 24, as, e.g., the extended cell shaded with gray consists of cells, which have either two or three reachable cells.

We define the partial order on the set of locations abstracting extended cells as follows.

Definition 65 (Partial Order of Locations)**.** Given a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S}) = (E, E_0, C, \Sigma, I, \Delta)$. Let $R_{\text{par}} \subseteq E \times E$ be a relation defined by

$$(e_{\text{ex},g}, e_{\text{ex},g'}) \in R_{\text{par}} \quad \text{iff} \tag{6.37a}$$

$$g_i \leq g'_i \quad \forall i \in \boldsymbol{k}. \tag{6.37b}$$

We write $e_{\text{ex},g} \sqsubseteq e_{\text{ex},g'}$ when $(e_{\text{ex},g}, e_{\text{ex},g'}) \in R_{\text{par}}$.

The relation R_{par} is a partial order relation, and $(E, \sqsubseteq)$ is a partially ordered set, which justifies its name.

From the partial order, we can provide a rough estimate of whether a location is reachable or not according to the following.

Proposition 25*: Given a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$, a partially ordered set $(E, \sqsubseteq)$ of its locations, and an initial location $e_{\text{ex},g}$. Let $e_{\text{ex},g} \sqsubseteq e_{\text{ex},g'}$, then*

$$\text{Reach}_{[0,\infty)}(\mathcal{A}, e_{\text{ex},g}) \cap e_{\text{ex},g'} = \emptyset. \tag{6.38}$$

We define a lattice according to the following.

Definition 66 (Lattice)**.** A partially ordered set $(X, \sqsubseteq)$ is said to be a lattice if for any finite set $X' \subseteq X$, the supremum and infimum of X' exist and belong to X.

Proposition 26*: Let $\mathcal{A}_{\text{ex}}(\mathcal{S}) = (E, E_0, C, \Sigma, I, \Delta)$ be a timed automaton generated from Procedure 1, and let $\sqsubseteq$ be the partial order from Definition 65. Then the partially ordered set $(E, \sqsubseteq)$ is a lattice.*

We see that $(E, \sqsubseteq)$ is a lattice by construction, as whenever, e.g., $e_{\text{ex},(a,b)}, e_{\text{ex},(b,a)} \in E$, then also $e_{\text{ex},(a,a)}, e_{\text{ex},(b,b)} \in E$, and $e_{\text{ex},(a,a)} < e_{\text{ex},(a,b)}$, $e_{\text{ex},(a,a)} < e_{\text{ex},(b,a)}$ and

$e_{\text{ex},(a,b)} < e_{\text{ex},(b,b)}$, $e_{\text{ex},(b,a)} < e_{\text{ex},(b,b)}$. Therefore, both supremum and infimum are defined for all $E' \subseteq E$.

Additionally, it is seen that almost all solutions of the system converge towards the location $\inf(E)$. By this we mean that the solutions end in the location $\inf(E)$. This can happen when a system with saddle points is abstracted. Consider a two-dimensional linear system with a saddle point. For this system all solutions initialized on one of the eigenaxis converge to the saddle point, which will not be abstracted by the location $\inf(E)$. However, in $\mathbb{R}^n$ for $n > 1$ this eigenaxis has Lebesgue measure zero. Therefore, we say that almost all solutions diverge from the saddle point. This also means that the analysis of convergence using $\inf(E)$ is "blind" to solutions of measure zero.

6 Conditions for the Subdivision

The purpose of this section is to set up necessary and sufficient conditions for the subdivision of the state space to generate sound, complete, and refinable abstractions.

Sound and Complete Abstractions

A useful abstraction shall preserve safety. Therefore, we introduce sound and complete abstractions in the following [26]. A sound abstraction can verify safety, and a complete abstraction can both verify and falsify safety.

Definition 67 (Sound Abstraction)**.** Let $\Gamma = (X, f)$ be a dynamical system and suppose its state space X is subdivided by $E(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$. Let the initial states $X_0 = \bigcup_{i \in \mathcal{I}} e_i$ with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ with $E_0 = \{e_i \mid i = \mathcal{I}\}$ is said to be a sound abstraction of Γ on $[t_1, t_2]$ if for all $t \in [t_1, t_2]$

$$e_i \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{6.39a}$$

$$\exists e_0 \in T_0 \text{ such that}$$

$$\alpha_E(e_i) \in \phi_{\mathcal{A}}(t, e_0). \tag{6.39b}$$

If a sound abstraction $\mathcal{A}$ is safe then Γ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$.

Definition 68 (Complete Abstraction)**.** Let Γ be a dynamical system and suppose its state space X is subdivided by $E(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$ and let the initial states be $X_0 = \bigcup_{i \in \mathcal{I}} e_i$ with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ with $e_0 = \{e_i \mid i = \mathcal{I}\}$ is said to be a complete abstraction of Γ on $[t_1, t_2]$ if it is a sound abstraction and for all $t \in [t_1, t_2]$ and

$$\text{for each } e_i \in \text{Reach}_{[t,t]}(\mathcal{A}, E_0) \tag{6.40a}$$

$$\exists x_0 \in X_0 \text{ such that}$$

$$\phi_\Gamma(t, x_0) \in \alpha_E^{-1}(e_i). \tag{6.40b}$$

If a complete abstraction $\mathcal{A}$ is safe (unsafe) then Γ is also safe (unsafe).

A sound and a complete abstraction of a dynamical system is illustrated in Figure 6.11.

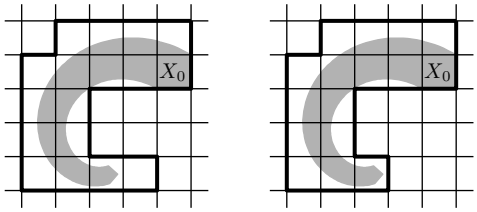


Figure 6.11: Illustration of the reachable set of a dynamical system (gray) from initial set X_0 and a sound approximation of this (cells within bold black lines) on the left and a complete abstraction on the right. Note that the lowest right cell is not reached by the dynamical system, but is reached by the sound abstraction.

Remark 12*: It is not sufficient to demand that an abstraction is complete, since an abstraction with only one location abstracting the entire state space is always complete.*

Proposition 27*: A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S}) = \mathcal{A}_1 || \ldots || \mathcal{A}_k$ is a sound (complete) abstraction of the system Γ, if and only if $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are sound (complete) abstractions of Γ.*

Sufficient conditions for soundness and completeness of an abstraction are formulated in the following.

Proposition 28 (Sufficient Condition for Soundness)*: A timed automaton $\mathcal{A}(\mathcal{S})$ is a sound abstraction of the system Γ, if its invariants and guards satisfy*

$$\underline{t}_{S^i_{g_i}} \leq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{\sup\{|L_f\varphi^i(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{g_i}\}} \tag{6.41a}$$

$$\bar{t}_{S^i_{g_i}} \geq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{\inf\{|L_f\varphi^i(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{g_i}\}} \tag{6.41b}$$

where $L_f\varphi^i(x)$ is defined as shown in (6.4a).

The values of guards of $\underline{t}_{S^i_{g_i}}$ and $\bar{t}_{S^i_{g_i}}$ can be algorithmically generated. The invariant $\bar{t}_{S^i_{g_i}}$ can be generated from Algorithm 1 and the guard can be generated from Algorithm 2.

Algorithm 1*: Suppose $\Gamma = (X, f)$ is a linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Let $L_f\varphi^i = -x^{\mathrm{T}} Q^i x$, and define $\underline{\gamma}^i$ as the solution to the following optimization problem*

$$\text{maximize } \underline{\gamma}^i \tag{6.42a}$$

$$\text{subject to } \underline{\gamma}^i P^i - Q^i \leq 0, \qquad \underline{\gamma}^i > 0. \tag{6.42b}$$

Then for any location $e_{(g,h)} \in E$, the invariant is

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \overline{t}_{S^i_{g_i}}, \text{ where} \tag{6.43a}$$

$$\overline{t}_{S^i_{g_i}} \geq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{a^i_{g_i-1}\underline{\gamma}^i}. \tag{6.43b}$$

Algorithm 2: Suppose $\Gamma = (X, f)$ is a linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Let $L_f\varphi = -x^{\mathrm{T}} Q^i x$ and define $\overline{\gamma}^i$ to be the solution to the following optimization problem

$$\min \overline{\gamma}^i \text{ subject to} \tag{6.44a}$$

$$Q^i - \overline{\gamma}^i P^i \leq 0 \tag{6.44b}$$

$$\overline{\gamma}^i > 0. \tag{6.44c}$$

Then for every pair of locations, $e_{(g,h)}, e_{(g',h')} \in E$, where $g'_i - 1 = g_i$ there is a transition relation

$$\delta_{(g,h)\to(g',h')} = (e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, e_{(g',h')}),$$

where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{6.45a}$$

and

$$\underline{t}_{S^i_{g_i}} \leq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{a^i_{g_i}\overline{\gamma}^i}. \tag{6.45b}$$

The sufficient condition states that the abstraction is sound if $\underline{t}_{S^i_{g_i}}$ is less than or equal to the time it takes to traverse $S^i_{g_i}$ maintaining a constant speed equal to the largest possible speed within $S^i_{g_i}$. Similarly, $\overline{t}_{S^i_{g_i}}$ should be greater than or equal to the time it takes to traverse $S^i_{g_i}$ maintaining a constant speed equal to the smallest possible speed within $S^i_{g_i}$. In the next example of a one-dimensional system, we calculate these sufficient conditions.

Example 10. Consider the system $\Gamma = (X, f)$, where f is

$$\dot{x} = x \tag{6.46}$$

and $X = [0, 3]$. Let $\varphi = \frac{1}{2}x^2$ be a subdivisioning function for the system. Then $L_f\varphi = -x^2$ according to (6.4a). An illustration of the subdivisioning function φ and its derivative $L_f\varphi$ is shown in Figure 6.12. We subdivide X into three slices utilizing the set of values $\left(a^1_1, a^1_2, a^1_3\right) = (1/2, 2, 9/2)$ such that $S^1_1 = [0, 1]$, $S^1_2 = [1, 2]$, and $S^1_3 = [2, 3]$, as shown in Figure 6.13.

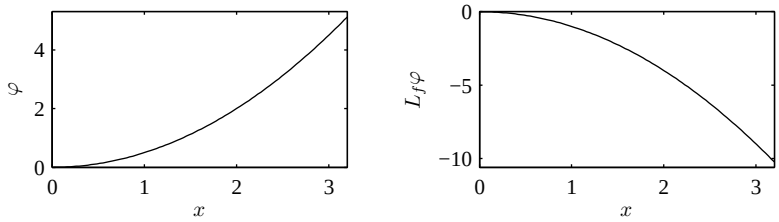


Figure 6.12: Plot of the subdivisioning function and its derivative.

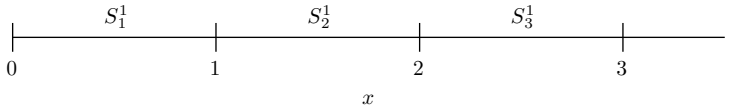


Figure 6.13: Illustration of a one-dimensional state space divided into three slices.

Now, it is possible to calculate the sufficient conditions for soundness shown in (6.41). These are

$$\underline{t}_{S_2^1} \leq \frac{|a_2^1 - a_1^1|}{|-(a_2^1)^2|} = \frac{3/2}{4} \text{ s} \qquad \bar{t}_{S_2^1} \geq \frac{|a_2^1 - a_1^1|}{|-(a_1^1)^2|} = 3/2 \text{ s} \tag{6.47a}$$

$$\underline{t}_{S_3^1} \leq \frac{|a_3^1 - a_2^1|}{|-(a_3^1)^2|} = \frac{5/2}{9} \text{ s} \qquad \bar{t}_{S_3^1} \geq \frac{|a_3^1 - a_2^1|}{|-(a_2^1)^2|} = \frac{5/2}{4} \text{ s}. \tag{6.47b}$$

Note that neither guards nor invariants for S_1^1 are calculated, as the location associated with this slice has no outgoing transitions and we can stay in this location forever.

Proposition 29 (Sufficient Condition for Completeness)*: Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families, and let*

$$S_{g_i}^i = (\varphi^i)^{-1}([a_{g_i-1}^i, a_{g_i}^i]). \tag{6.48}$$

A deterministic timed automaton $\mathcal{A}(\mathcal{S})$ is a complete abstraction of Γ if

1. *for any $g \in \mathcal{G}(\mathcal{S})$, recall the definition of $\mathcal{G}(\mathcal{S})$ from Definition 57, with $g_i \geq 2$ there exists a time $t_{g_i}^i$ such that for all $x_0 \in (\varphi^i)^{-1}(a_{g_i}^i)$*

$$\phi_\Gamma(t_{g_i}^i, x_0) \in (\varphi^i)^{-1}(a_{g_i-1}^i) \tag{6.49}$$

   and

2. $\bar{t}_{S_{g_i}^i} = \underline{t}_{S_{g_i}^i} = t_{g_i}^i$.

Equation (6.49) states that it takes the time $t_{g_i}^i$ for all trajectories of Γ to propagate from $(\varphi^i)^{-1}(a_{g_i}^i)$ to $(\varphi^i)^{-1}(a_{g_i-1}^i)$ (i.e., $t_{g_i}^i$ is the time to travel through slice $S_{g_i}^i$). Utilizing this time for both the invariant and guard conditions (i.e., $\bar{t}_{S_{g_i}^i} = \underline{t}_{S_{g_i}^i} = t_{g_i}^i$) implies that the abstraction is complete. Recall that $\bar{t}$ is used for invariants, while $\underline{t}$ is used for guard conditions.

Example 11. Consider the system defined in Example 10 and subdivide it into the same slices S_1^1, S_2^1, and S_3^1.

The condition shown in (6.49) is automatically satisfied, as there is only one solution from each level surface of φ. Therefore, $\underline{t} = \overline{t}$, i.e.,

$$\underline{t}_{S_2^1} = \overline{t}_{S_2^1} = \ln(4)\ \text{s} \tag{6.50a}$$

$$\underline{t}_{S_3^1} = \overline{t}_{S_3^1} = \ln(9/4)\ \text{s}. \tag{6.50b}$$

This finalizes the explanation of sound and complete abstractions, but as stared in Remark 12 another condition, called refinability, should also be considered.

Refinable Abstraction

To ensure that an abstraction can get any desired accuracy, it should be refinable according to the following definitions.

Definition 69 (Refinement of Subdivision)**.** Let the subdivision $E(\mathcal{S})$ be generated by the slice-families $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then the subdivision $E(\tilde{\mathcal{S}})$ is a refinement of $E(\mathcal{S})$ if and only if for any $e \in E(\mathcal{S})$ there is a cell $\tilde{e} \in E(\tilde{\mathcal{S}})$ such that

$$\tilde{e} \subseteq e. \tag{6.51}$$

Definition 70 (Refinable Abstraction)**.** An abstraction $\mathcal{A}(\mathcal{S})$ of a system Γ is said to be refinable if for all $\epsilon > 0$ there exists an abstraction $\mathcal{A}(\tilde{\mathcal{S}})$, where $E(\tilde{\mathcal{S}})$ is a refinement of $E(\mathcal{S})$, such that for all $\tilde{e} \in E(\tilde{\mathcal{S}})$

$$\tilde{e} \subseteq B(\epsilon), \tag{6.52}$$

where $B(\epsilon)$ is a ball with radius ϵ.

The least ϵ that satisfies (6.52) for all $\tilde{e} \in K(\tilde{\mathcal{S}})$ is called the radius of the subdivision. We see that combining Definition 68 and Definition 70 yields the following corollary.

Corollary 6*: If the system Γ is abstracted with a complete and refinable abstraction $\mathcal{A}(\mathcal{S})$, then for all $\epsilon > 0$ there exists a subdivision $\mathcal{A}(\tilde{\mathcal{S}})$ such that for all $t \in [t_1, t_2]$*

$$\alpha_K^{-1}\left(\text{Reach}_{[t,t]}(\mathcal{A}(\tilde{\mathcal{S}}), E_0)\right) \subseteq \text{Reach}_{[t,t]}(\Gamma, X_0) + B(\epsilon). \tag{6.53}$$

We say that the accuracy of the abstraction is the smallest ϵ such that (6.53) is satisfied. Corollary 6 states that a complete and refinable abstraction can approximate the reachable states of a system Γ arbitrarily close; hence, this type of abstraction solves Problem 9 in Section 1. In conclusion, to get any desired radius of the subdivision, all cells of its subdivision $E(\mathcal{S})$ should converge towards points. In the next proposition, we answer the question of minimal number of slice-families necessary to construct a refinable abstraction.

Proposition 30 (Necessary Condition for Refinable Abstraction)*: If $\mathcal{A}(\mathcal{S})$ is a refinable abstraction of a system Γ, then $\mathcal{S}$ is a collection of n slice-families.*

To clarify this proposition the following example is provided.

Example 12. Consider a subdivision of the state space of a two-dimensional system generated utilizing one slice-family shown in Figure 6.14.

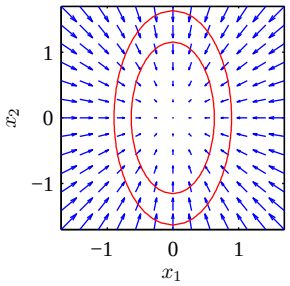


Figure 6.14: Subdivision of a two-dimensional state space utilizing one slice-family.

Imagine that level curves are added to improve the accuracy of the subdivision. It is seen that the smallest cells that can be created utilizing only one slice-family are level curves of the subdivisioning function. Therefore, the radius of the subdivision cannot converge to zero, as required by Definition 70. However, subdividing the same state space utilizing two slice-families would allow the radius of the subdivision to go to zero.

Now, the conditions for obtaining sound, complete, and refinable abstractions have been set up. In the next section, a method for synthesizing such abstractions using Lyapunov functions is provided.

7 Subdividing the State Space using Lyapunov Functions

To synthesize the subdivision presented in this paper, we need to generate subdivisioning functions. As subdivisioning functions, we use Lyapunov functions since their sub-level sets are positive invariant sets. It is beneficial to use Lyapunov functions as subdivisioning functions, as this enables the use of existing methods for generating Lyapunov functions used in controller design. This is possible even though Lyapunov functions are usually associated to systems with stable equilibria, but recall from Definition 43 that in this context they are associated to dynamical systems with hyperbolic equilibria. This is done to allow Lyapunov functions to be subdivisioning functions for both stable and unstable systems.

In the following, we provide existence results for refinable, sound, and complete abstractions.

Refinable Abstraction

The use of continuous subdivisioning functions, see Definition 54, for generating the subdivision gives a natural and easy way to describe the boundaries of slices and cells. This also automatically makes subdivisions generated by n slice-families refinable.

A subdivision generated by continuous subdivisioning functions can be refined, since for any slice $\varphi^{-1}([a_1, a_2])$ the regular values a_1 and a_2 can be chosen arbitrarily close to

each other. Therefore, if n such functions exist, the abstraction generated from the subdivision is refinable. The existence of n Lyapunov functions for a Morse-Smale system is provided in Proposition 31.

Sound Abstraction

In this subsection we show that there exists a sound abstraction of any Morse-Smale system. We do this by showing the existence of Lyapunov functions for Morse-Smale systems, actually we show that there exists n Lyapunov functions for any Morse-Smale system; hence, they are sound and refinable abstractions.

Definition 71. Let $a \in \mathrm{CrV}(f)$ if and only if there is $p \in Cr(f)$ such that $f(p) = a$ and suppose $\Gamma = (X, f)$. Then two Lyapunov functions $\varphi^1, \varphi^2 : \mathbb{R}^n \to \mathbb{R}$ are transversal if the level sets $(\varphi^1)^{-1}(a)$ and $(\varphi^2)^{-1}(a)$ are transversal for any $a \in \mathbb{R} \backslash \{\mathrm{CrV}(f)\}$.

Proposition 31*: Let $n > 1$. For any Morse-Smale system on $\mathbb{R}^n$, there exists n transversal Lyapunov functions.*

From this, the following theorem follows easily.

Theorem 10*: Let $\Gamma = (X, f)$ be a Morse-Smale system and let $n > 1$. Then there exists a sound and refinable abstraction $\mathcal{A}(\mathcal{S})$ of Γ generated by n transversal Lyapunov functions.*

Complete Abstraction

To generate complete abstractions, we set up a proposition that gives an easy testable condition for completeness. A complete abstraction of (6.1) can be obtained by constructing a subdivision generated by Lyapunov functions, which satisfies Proposition 29.

Proposition 32*: Let each slice-family of $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be associated with a Lyapunov function $\varphi^i(x)$ for the system Γ, such that $S^i_j = (\varphi^i)^{-1}([a^i_{j-1}, a^i_j])$ and let*

$$\dot{\varphi}^i(x) = \alpha L_f \varphi^i(x) \qquad \forall x \in \mathbb{R}^n. \tag{6.54}$$

Then $\mathcal{A}(\mathcal{S})$ is a complete abstraction of Γ.

The existence of such Lyapunov functions is addressed in the following proposition.

Proposition 33*: For any hyperbolic linear system Γ there exists n transversal Lyapunov functions $\varphi_i(x)$ each satisfying*

$$\dot{\varphi}^i(x) = \alpha L_f \varphi^i(x) \qquad \forall x \in \mathbb{R}^n. \tag{6.55}$$

Theorem 11*: For any hyperbolic linear system Γ there exists a complete abstraction given by n transversal Lyapunov functions.*

8 Examples

To illustrate the use of the proposed abstraction method, three examples are provided in this section. In the first example, we provide the reachable sets of a sound and a

complete abstractions of a simple dynamical system are provided. In the second example, we demonstrate what type of questions we can answer using the proposed abstraction. Note that these requirements cannot be verified using conventional technics from control theory; however, using some existing simulation-based verification tools the verification is possible. In the third example, we demonstrate that the method is applicable for high-dimensional systems, by verifying a 100 dimensional system.

In the following, we compare the reachable sets of a sound and a complete abstraction and see how their different subdivisions look. It is chosen to utilize a two-dimensional linear system in the example, since it is possible to visualize its state space, but the method applies for systems of arbitrary dimension. The considered linear system is specified in the following

$$\dot{x} = Ax \tag{6.56}$$

with $X = [-10, 10] \times [-10, 10]$

$$A = \begin{bmatrix} -3 & -1 \\ -2 & -5 \end{bmatrix} \text{ and } X_0 = [1.5, 2] \times [-10, -9.5].$$

In both cases of the sound and the complete abstraction, the state space is subdivided utilizing two different Lyapunov functions, i.e., two slice-families. Furthermore, each slice-family consists of 10 sub-level sets, which are equally distributed on the considered state space Figure 6.15. Both abstractions are timed automata with 361 locations (100 extended locations).

To simplify Figure 6.15, we have chosen to only depict the subdivisions of the state space, not the timed automata.

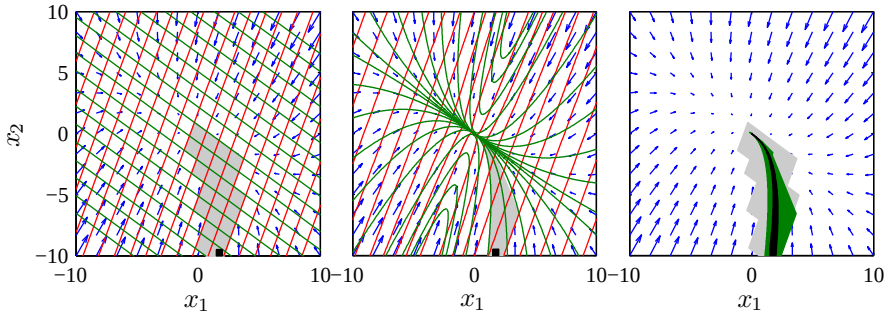


Figure 6.15: Illustration of a sound abstraction (left subplot), a complete abstraction (middle subplot), and a comparison of their reachable sets (right subplot). The black area in the right subplot is the reachable state space of (6.56), the green area is the reachable state space of the complete abstraction, and the gray area is the reachable states of the sound abstraction.

In the figure, the reachable states of the two abstractions are shaded gray and are compared with the reachable space of the considered system in the right subplot. Notice that no effort is done to make the initial states (black box) fit with the cells of the subdivisions; hence, this introduces some over-approximation of the abstractions. In conclusion,

the reachable state space for the sound abstraction is larger than the reachable state space for the complete abstraction, as expected.

When we look at the shape of the cells of the two subdivisions, it is seen that the cells of the complete abstraction look more complicated than the shape of the sound subdivision. Additionally, more computations are necessary for generating the complete abstraction, as it is generated via conjugate transformation, as explained in the proof of Proposition 33. Hence, the increased accuracy comes at a price - increased computational complexity.

The next example shows what type of specifications we are capable of checking using the proposed abstraction. In the example, we consider a slightly more complicated example, and a very complicated specification. The system is given by the following three-dimensional system (again the dimensionality is kept low to allow visualization of the example)

$$\dot{x} = \begin{bmatrix} -0.1 & -1 & 0 \\ 1 & -0.1 & 0 \\ 0 & 0 & -0.15 \end{bmatrix} x. \tag{6.57}$$

Now, we check if the system satisfies the following specification, which is illustrated in Figure 6.16: Does all trajectories of the system (6.57) initialized in $X_0 = [-0.1,\, 0.1] \times [0.8,\, 1] \times [0.9,\, 1]$ (blue box)

- avoid the unsafe region (red box),
- reach the bottom half of the state space (below the gray surface) within 10 s,
- and reach the goal set (green box) within 20 s and stay there.

To verify this specification, we subdivide the state space using three quadratic Lyapunov functions $\varphi^i(x) = x^{\mathrm{T}} P^i x$, for $i \in \{1, 2, 3\}$ and

$$P^1 = \begin{bmatrix} 0.0050 & 0 & 0 \\ 0 & 0.0050 & 0 \\ 0 & 0 & 3.3333 \end{bmatrix} \tag{6.58}$$

$$P^2 = \begin{bmatrix} 4.3069 & 0.0693 & 0 \\ 0.0693 & 4.6931 & 0 \\ 0 & 0 & 0.0033 \end{bmatrix} \tag{6.59}$$

$$P^3 = \begin{bmatrix} 5.2475 & -0.0248 & 0 \\ -0.0248 & 4.7525 & 0 \\ 0 & 0 & 0.0033 \end{bmatrix}. \tag{6.60}$$

The subdivision is not shown in the figure; whereas, both the requirements and a set of trajectories of (6.57) are illustrated.

The analysis of the system says that the requirements are satisfied, as no trajectories reach the red box, all trajectories go below $x_3 = 0.5$ within 7 s and stay inside the green box after 12.7 s. This also complies with the simulated trajectories shown in Figure 6.16.

Now, we verify a 100-dimensional system, by checking if all trajectories initialized in the unit hypercube centered at x_{init} avoid the unit hypercube centered at x_{avoid}. We do not want to write all the 10,000 elements of the system matrix and the 100 coordinates of x_{init}

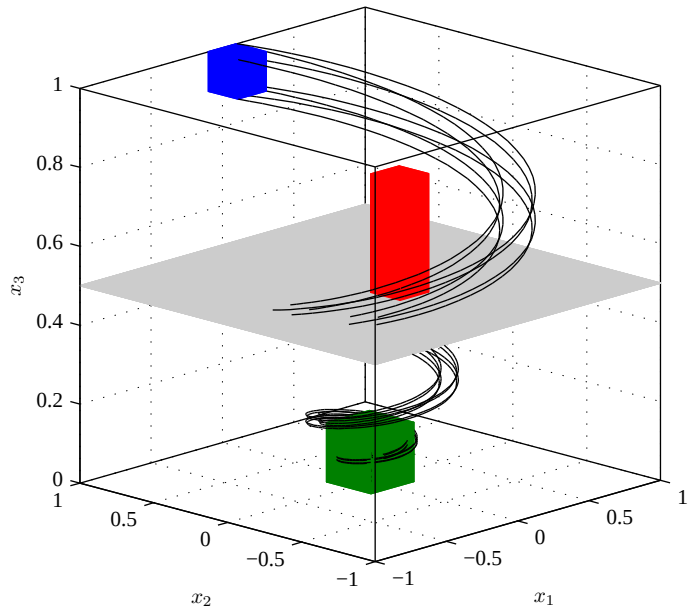


Figure 6.16: State space of the considered system. The blue box illustrates the initial states of the system, the red box illustrates the unsafe states, and the green box illustrated the goal states. The plane given by $x_3 = 0.5$ is illustrated with gray and finally a set of system trajectories are drawn with black lines.

and x_{avoid}. Therefore, we provide the MATLAB code necessary to reproduce the setup in the following:

```
rand('state',1); \% Set state of random generator
randn('seed',1); \% Set seed of random generator
SYS = rss(100,0); \% Generate 100 dimensional random state space model
rand('state',2); \% Set state of random generator
randn('seed',2); \% Set seed of random generator
x_init = 10*rand(1,100);\% Pick 100 random numbers from uniform dist.
rand('state',3); \% Set state of random generator
randn('seed',3); \% Set seed of random generator
x_avoid = 8*rand(1,100); \% Pick 100 random numbers from uniform dist.
```

Before verifying a system of this size, it would be appropriate to apply some model order reduction techniques [27] to obtain a lower order model. However, to demonstrate that it is possible to verify such systems by the proposed method, the dimension is kept at 100; hence, we use 100 Lyapunov functions.

It is not reasonable to just generate one automaton with a location for each cell of the subdivision, as the automaton would have billions of locations. Therefore, we generate one timed automaton per Lyapunov function (i.e. 100 timed automata). The automata are analyzed separately, giving a time interval in which a solution may be in x_{avoid}. If the conjunction of all these time intervals is empty, then no trajectory reaches x_{avoid}. In the considered example, this time interval is empty; hence, all trajectories avoid x_{avoid}.

9 Conclusion

In this paper, a method for abstracting dynamical systems by timed automata is proposed. The abstraction is based on subdividing the state space of the dynamical systems by set-differences of positively invariant sets.

To enable both verification and falsification of safety of the considered system based on the abstraction, conditions for soundness, completeness, and refinability are derived. Furthermore, it has been demonstrated that the abstraction can be obtained as a parallel composition of multiple timed automata under certain conditions.

Via algorithms, based on LMI-based optimization problems, it is shown how the conditions for the abstraction can be checked and how time information for the timed automaton can be generated for linear systems.

It is shown that there exist sound and refinable abstractions for hyperbolic Morse-Smale systems and complete and refinable abstractions for all hyperbolic linear systems. Finally, an example of a sound and a complete abstraction is provided and their reachable sets are compared.

In the presented work, it is seen that abstractions generated by subdividing the state space of a system with the use of Lyapunov functions can be designed to be sound, complete, and refinable. Furthermore, a priori an upper bound on the over-approximation of the reachable set introduced by the abstraction can be calculated. Finally, an example shows that the method is applicable for high-dimensional systems, due to its compositionality.

A Definitions

Definition 72 (Morse-Smale System [28])**.** A smooth vector field $X \in \mathfrak{X}^r(M)$ will be called a Morse-Smale system (or field) provided it satisfies the following conditions:

1. X has a finite number of singular points, say $\beta_1, \ldots, \beta_k$, each hyperbolic. A hyperbolic singular point is a singular point such that in local coordinates the matrix of partial derivatives of X has eigenvalues with nonzero real parts.
2. X has a finite number of closed orbits (periodic solutions), say $\beta_{k+1}, \ldots, \beta_n$, each hyperbolic.
3. For any $p \in M$, $\alpha(p) = \beta_i$ and $\omega(p) = \beta_j$ for some i and j.
4. Let $\Omega(X)$ be the nonwandering[1] points for X, then $\Omega(X) = \{\beta_1, \ldots, \beta_N\}$.
5. The stable and unstable manifolds associated with the β_i have transversal intersection.

Definition 73 (Topologically Equivalent Vector Fields [29])**.** Two vector fields $\xi, \eta \in \mathfrak{X}^r(M)$ are topologically equivalent if there exists a homeomorphism $h : M \to N$ (h is continuous and has continuous inverse) such that

1. $h \circ \phi_\xi(\mathbb{R}, x_0) = \phi_\eta(\mathbb{R}, h(x_0))$ for each $x_0 \in M$,

[1] We say that $p \in M$ is a wandering point for X if there exists a neighborhood V of p and a number t_0 such that $\phi_X(t, V) \cap V = \emptyset$ for $|t| > t_0$.

2. h preserves the orientation, that is if $x_0 \in M$ and $\delta > 0$ there exists $\epsilon > 0$ such that, for $0 < t < \delta$, $h \circ \phi_\xi(t, x_0) = \phi_\eta(\tau, h(x_0))$ for some $0 < \tau < \epsilon$.

Remark 13: *Two vector fields are topologically conjugate if $t = \tau$ in the previous definition.*

From Definition 73, we see that the solution of a vector field η from some initial state can be described by a continuous deformation (the homeomorphism h) of the solution to a topologically equivalent vector field ξ. For a more formal explanation, see Proposition 4.1 in [18].

Definition 74 (Regular Value [24])**.** Let $f : N \to M$ be a smooth map. A point p in N is a regular point if the differential

$$f_{*,p} : T_pN \to T_{f(p)}M \tag{6.61}$$

is surjective. A point in M is a regular value if it is the image of a regular point.

Definition 75 (Timed-Abstracted Bisimulation [22])**.** Let $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ be a timed automaton. A reflexive and symmetric relation $R \subseteq E \times \mathbb{R}^C \times E \times \mathbb{R}^C$ is a time-abstracted bisimulation if for all $(e_1, v_1)R(e_2, v_2)$, (Note that we denote $(e_1, v_1, e_2, v_2) \in R$ by $(e_1, v_1)R(e_2, v_2)$)

- for all $(e_1, v_1) \xrightarrow{d} (e_1', v_1')$ there exists $(e_1', v_1')R(e_2', v_2')$ for which $(e_2, v_2) \xrightarrow{d} (e_2', v_2')$, and
- for all $(e_1, v_1) \xrightarrow{\sigma} (e_1', v_1')$, $\sigma \in \Sigma$, there exists $(e_1', v_1')R(e_2', v_2')$ for which $(e_2, v_2) \xrightarrow{\sigma} (e_2', v_2')$.

We say $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimilar if there exists a time-abstracted bisimulation R for $(\mathcal{A}_1, \mathcal{A}_2)$.

B Proofs

Proposition 18. If $e_{(g,h)} \bigcap (\varphi^i)^{-1}(a^i_{g_i - 1})$ is not connected for some i, then σ^i is the label of multiple outgoing transitions from the location $e_{(g,h)}$, i.e., there exist multiple transitions in Δ, where $e_{(g,h)}$ is the source location and σ^i is the label. Therefore, the timed automaton $\mathcal{A}(\mathcal{S})$ is nondeterministic. □

Proposition 21. Consider the timed automaton $\mathcal{A}_{||}(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1)|| \dots ||\mathcal{A}_k(\mathcal{S}^k)$ where $\mathcal{A}_i(\mathcal{S}^i) = (E_i, E_{0,i}, C_i, \Sigma_i, I_i, \Delta_i)$ and $E_i = \{e^i_1, \dots, e^i_{|\mathcal{S}_i|}\}$, abstracting the slices $S^i_1, \dots, S^i_{|\mathcal{S}_i|}$. Then the timed automaton $\mathcal{A}_{||}(\mathcal{S})$ is given by

- **Locations:** $E = E_1 \times \cdots \times E_k$, which according to Definition 57 represents extended cells, if the transversal intersection of all slices is nonempty i.e. (6.31) is satisfied.
- **Clocks:** $C = \{c^i, \dots, c^k\}$, where c^i monitors the time for being in a slice of $\mathcal{S}^i$.

- **Invariants:** The invariant for location $e_{\text{ex},g} = (e^1_{g_1}, \dots, e^k_{g_k})$ is identical to (6.22) and is

$$I(e_{\text{ex},g}) = \bigwedge_{i=1}^{k} I_i(e^i_{g_i}). \tag{6.62}$$

- **Input Alphabet:** $\Sigma = \{\sigma^1, \dots, \sigma^k\}$.
- **Transition relations:** Σ_i is disjoint from Σ_j for all $i \neq j$; hence, item 1) in Definition 63 never happens.

This implies that $\mathcal{A}_{||}(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1)|| \dots ||\mathcal{A}_k(\mathcal{S}^k)$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are isomorph. □

Proposition 23. If the locations of $\mathcal{A}_{\text{ex}}$ are extended cells, then soundness of $\mathcal{A}_{\text{ex}}$ can be reformulated to the following.

A timed automaton $\mathcal{A}_{\text{ex}}$ with $E_0 = \{e_{\text{ex},g} | g \in \mathcal{G}_0 \subseteq \mathcal{G}\}$ is said to be a sound abstraction of Γ with $X_0 = \bigcup_{g\in\mathcal{G}_0} e_{\text{ex},g}$ on $[t_1, t_2]$ if for all $t \in [t_1, t_2]$ and for all $g \in \mathcal{G}$

$$\bigcap_{i=1}^{k} S^i_{g_i} \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{6.63a}$$

$$\exists e_0 \in E_0 \text{ such that}$$

$$\bigcap_{i=1}^{k} S^i_{g_i} \in \alpha_K^{-1}(\phi_{\mathcal{A}_{\text{ex}}}(t, e_0)) \tag{6.63b}$$

which is equivalent to: For all $i \in \boldsymbol{k}$, all $g \in \mathcal{G}$, and for all $t \in [t_1, t_2]$

$$S^i_{g_i} \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{6.64a}$$

$$\exists e_{0,i} \in E_{0,i} \text{ such that}$$

$$\alpha_K^{-1}(\phi_{\mathcal{A}_i}(t, e_{0,i})) = S^i_{g_i}. \tag{6.64b}$$

From (6.64) it is seen that $\mathcal{A}_{\text{ex}} = \mathcal{A}_1|| \dots ||\mathcal{A}_k$ is sound if and only if $\mathcal{A}_i$ is sound for $i \in \boldsymbol{k}$. Similar arguments can be used to prove the completeness part of Proposition 27. □

Proposition 24. Let $e_{(g,h)}$ with $h \in \boldsymbol{m}$ be the cells which union is the extended cell $e_{\text{ex},g}$. Then

$$I(e_{(g,h)}) = I(e_{(g,k)}) \qquad \forall h, k \in \boldsymbol{m} \tag{6.65}$$

as the invariants are calculated based on slices (6.22).

If the subdivision satisfies (6.36), then the same outgoing transitions exist for all cells within the same extended cell. Furthermore,

$$G_{(g,h)\to(g',h')} = G_{(g,k)\to(g',k')} \qquad \forall h, k \in \boldsymbol{m} \tag{6.66}$$

since the guards are also calculated based on slices (19b) in [13]. This implies that all possible behaviors from each cell in an extended cell are the same; hence, $\mathcal{A}(\mathcal{S})$ is bisimilar to a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$. □

Proposition 28. Let $\mathcal{A}(\mathcal{S})$ be a timed automaton with $E_0 = \{e_i | i \in \mathcal{I}\}$, be an abstraction of Γ with initial set $X_0 = \bigcup_{i\in\mathcal{I}} e_i$. If guards and invariants of $\mathcal{A}(\mathcal{S})$ satisfy (6.41), then

$$\text{Reach}_{[t_1,t_2]}(\Gamma, X_0) \subseteq \alpha_K^{-1}(\text{Reach}_{[t_1,t_2]}(\mathcal{A}, E_0)) \tag{6.67}$$

since for all $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$ there exists $t \in [\underline{t}_{S^i_{g_i}}, \overline{t}_{S^i_{g_i}}]$ such that

$$\phi_\Gamma(t, x_0) \in (\varphi^i)^{-1}(a^i_{g_i-1}). \tag{6.68}$$

□

Proposition 29. The proposition states that it takes the same time for all trajectories of Γ to propagate between any two level sets of φ^i. From this it follows that $\mathcal{A}(\mathcal{S})$ is complete if $\overline{t}_{S^i_{g_i}}$ and $\underline{t}_{S^i_{g_i}}$ are equal to $t^i_{g_i}$. □

Proposition 30. If $\mathcal{A}(\mathcal{S})$ is a refinable abstraction, then for any $\epsilon > 0$ there exists a subdivision $E(\mathcal{S})$ such that (30) in [13] holds for cells in $E(\mathcal{S})$. Therefore,

$$S^i_{g_i} \subset (\varphi^i)^{-1}(a^i_{g_i}) + B(\epsilon) \tag{6.69}$$

where $\epsilon > 0$. Note that $a^i_{g_i}$ is a regular value of φ^i, i.e., the dimension of the level set $(\varphi^i)^{-1}(a^i_{g_i})$ is $n-1$. The locations of $\mathcal{A}(\mathcal{S})$ are cells for which

$$\bigcup_h e_{(g,h)} = \pitchfork_{i=1}^k S^i_{g_i} \tag{6.70a}$$

$$\subset \pitchfork_{i=1}^k \left(\varphi_i^{-1}(a^i_{g_i}) + B(\epsilon)\right) \tag{6.70b}$$

$$\subset \pitchfork_{i=1}^k \varphi_i^{-1}(a^i_{g_i}) + B(2\epsilon). \tag{6.70c}$$

But (6.70c) is true for any ϵ, thus it is enough to prove that

$$\dim\left(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})\right) = 0. \tag{6.71}$$

Using Theorem 7.7 in [30] the dimension of an extended cell is given by

$$\dim\left(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})\right) = [[(n-1) + (n-1) - n] + (n-1) - n] + (n-1) - n \ldots \tag{6.72a}$$

$$= k(n-1) - (k-1)n. \tag{6.72b}$$

We see that if $k \neq n$ then $\dim\left(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})\right) \neq 0$, thus we have contradiction. We conclude that $k = n$. □

Proposition 31. Let $S(n, \mathbb{R})$ be a set of $n \times n$ symmetric matrices. $S(n, \mathbb{R})$ is a subspace of $M(n, \mathbb{R})$ of $\dim(S(n, \mathbb{R})) = n(n+1)/2$. Consider the map $\varphi_A : S(n, \mathbb{R}) \to S(n, \mathbb{R})$ and let

$$P \mapsto A^{\mathrm{T}}P + PA. \tag{6.73}$$

Now consider the map $\det : M(n, \mathbb{R}) \to \mathbb{R}$ and let

$$A \mapsto \det(A). \tag{6.74}$$

Then $(\det \circ \varphi_A)^{-1}(\{0\})$ is a closed set. Therefore,

$$U_A \equiv \{P \in S(n,\mathbb{R}) | \det \circ \varphi_A(P) \neq 0\} \tag{6.75}$$

is an open set. $V_A \equiv V \cap U_A$ is open, where

$$V = \{P \in S(n,\mathbb{R}) | \det(P) \neq 0\} \qquad (V \text{ is open}). \tag{6.76}$$

Let $\Theta = \{Q \in S(n,\mathbb{R}) | Q > 0\}$ by Proposition 2.18 in [18] the map

$$M(n,\mathbb{R}) \to C^n/S^n \text{ defined by} \tag{6.77}$$

$$L \mapsto \text{diag}([\lambda_1, \ldots, \lambda_n]) \text{ is continuous.} \tag{6.78}$$

Thus Θ is an open set in $S(n,\mathbb{R})$.
We pick an open neighborhood around $Q = A^{\mathrm{T}}P + PA$ and denote it U. Then for every $Q' \in U$ there exists a (unique) P, thus $\varphi_A^{-1}(U)$ is a nonempty open set in $S(n,\mathbb{R})$.

We can pick n linear independent matrices $P_1, \ldots, P_n \in \varphi_A^{-1}(U)$. This is possible because $\varphi_A^{-1}(U)$ is open in $S(n,\mathbb{R})$ and $\dim(S(n,\mathbb{R}))$ is $n(n+1)/2$. Then for any $a \in \mathbb{R}\backslash\{0\}$ and $i \neq j$

$$\{x \in \mathbb{R}^n | x^{\mathrm{T}} P_i x = a\} \pitchfork \{x \in \mathbb{R}^n | x^{\mathrm{T}} P_j x = a\}. \tag{6.79}$$

Extending this to Morse-Smale systems follows directly from Theorem 1 in [16]. □

Proposition 32. Let φ be a Lyapunov function for the system Γ and let $x, x' \in \varphi^{-1}(a_m)$. According to Proposition 29 the abstraction is complete if there exists a t_m, for $m = 2, \ldots, k$ such that

$$\phi_\Gamma(t_m, x), \phi_\Gamma(t_m, x') \in \varphi^{-1}(a_{m-1}). \tag{6.80}$$

This is true if

$$L_f\varphi(\phi_\Gamma(t,x)) - L_f\varphi(\phi_\Gamma(t,x')) = 0 \qquad \forall t. \tag{6.81}$$

The combination of (6.80) and (6.81) implies that for all $c > 0$ there exists an α such that

$$\varphi^{-1}(c) = (L_f\varphi)^{-1}\left(\frac{c}{\alpha}\right) \tag{6.82}$$

hence for all x there exists an α such that

$$\varphi(x) = \alpha L_f\varphi(x). \tag{6.83}$$

□

Proposition 33. This is proved for linear systems, by constructing the complete abstraction.

Consider a linear differential equation

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}\lambda_1 I_1 & 0 \\ 0 & \lambda_2 I_2\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} \tag{6.84}$$

where I_1, I_2 are identity matrices and $\lambda_1 < 0$ and $\lambda_2 > 0$.

The stable and unstable subspaces of (6.84) are orthogonal and can be treated separately. This system is divided into a stable space described by x_1 and an unstable space described by x_2. For $i \in \{1, 2\}$ let $\varphi_i(x_i) = x_i^{\mathrm{T}} P_i x_i$ be a quadratic Lyapunov function. Then its derivative is $L_f\varphi(x_i) = x_i^{\mathrm{T}} Q_i x_i$, where

$$2\lambda_i P_i = Q_i \qquad \text{for } i = 1, 2. \tag{6.85}$$

This implies that any quadratic Lyapunov function satisfies Proposition 32 and hence generates a complete abstraction.

Since hyperbolic linear systems are topologically conjugate if and only if they have the same index [31]. There is a homeomorphism $h : \mathbb{R}^n \to \mathbb{R}^n$ such that any hyperbolic linear system is topologically conjugate of (6.84), by choosing I_1 and I_2 appropriately. Note that h is a diffeomorphism on $\mathbb{R}^n \backslash \{0\}$.

This implies that there exists a complete abstraction of every hyperbolic linear system. □

References

[1] E. Asarin, T. Dang, G. Frehse, A. Girard, C. L. Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design*, Munich, Germany, 2006, pp. 1582–1587.

[2] H. Guéguen, M.-A. Lefebvre, J. Zaytoon, and O. Nasri, "Safety verification and reachability analysis for hybrid systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 25–36, 2009.

[3] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3414, pp. 291–305.

[4] A. B. Kurzhanski and I. Vályi, *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser Boston, 1997.

[5] H. Yazarel and G. J. Pappas, "Geometric programming relaxations for linear system reachability," in *Proceedings of the 2004 American Control Conference*, Boston, MA, USA, 2004, pp. 553–559.

[6] O. Maler and G. Batt, "Approximating continuous systems by timed automata," in *Formal Methods in Systems Biology*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, vol. 5054, pp. 77–89.

[7] A. Tiwari, "Abstractions for hybrid systems," *Formal Methods in System Design*, vol. 32, no. 1, pp. 57—83, 2008.

[8] L. d. Alfaro, T. A. Henzinger, and R. Majumdar, "Symbolic algorithms for infinite-state games," in *Proceedings of the 12th International Conference on Concurrency Theory*, Aalborg, Denmark, 2001, pp. 536–550.

[9] M. Morse and G. A. Hedlund, "Symbolic dynamics," *American Journal of Mathematics*, vol. 60, no. 4, pp. 815–866, 1938.

[10] R. Ghosh and C. Tomlin, "Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-Notch protein signalling," *Systems Biology*, vol. 1, no. 1, pp. 170–183, June 2004.

[11] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on Uppaal," in *International School on Formal Methods for the Design of Real-Time Systems*, vol. 3185, 2004, pp. 200–237.

[12] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking for real-time systems," in *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, June 1990, pp. 414–425.

[13] C. Sloth and R. Wisniewski, "Abstraction of continuous dynamical systems utilizing Lyapunov functions," in *Proceedings of the 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, USA, December 2010, pp. 3760–3765.

[14] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 116–126, January 2010.

[15] F. Clarke, Y. Ledyaev, R. Stern, and P. Wolenski, *Nonsmooth Analysis and Control Theory*. Springer, 1998.

[16] K. R. Meyer, "Energy functions for Morse Smale systems," *American Journal of Mathematics*, vol. 90, no. 4, pp. 1031–1040, 1968.

[17] Y. Matsumoto, *An Introduction to Morse Theory*. American Mathematical Society, 2002.

[18] J. P. Junior and W. de Melo, *Geometric Theory of Dynamical Systems: An Introduction*. Springer, 1980.

[19] M. W. Hirsch, *Differential Topology*. Springer, Heidelberg, 1976.

[20] M. M. Peixoto, "Structural stability on two-dimensional manifolds," *Topology*, vol. 1, no. 2, pp. 101–120, 1962.

[21] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, April 1994.

[22] U. Fahrenberg, K. Larsen, and C. Thrane, "Verification, performance analysis and controller synthesis for real-time systems," in *Fundamentals of Software Engineering*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 5961, pp. 34–61.

[23] M. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, USA, December 1998, pp. 4277–4282.

[24] L. W. Tu, *An Introduction to Manifolds*. Springer, 2008.

[25] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. SIAM studies in applied mathematics. SIAM, 1994, vol. 15.

[26] R. Alur, T. Dang, and F. Ivančić, "Progress on reachability analysis of hybrid systems using predicate abstraction," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003, vol. 2623, pp. 4–19.

[27] A. C. Antoulas, *Approximation of large-scale dynamical systems*, ser. Advances in Design and Control. SIAM, 2005.

[28] R. Wisniewski, "Flow lines under perturbations within section cones," Ph.D. dissertation, Department of Mathematical Sciences, Aalborg University, 2005.

[29] R. Wisniewski and M. Raussen, "Geometric analysis of nondeterminacy in dynamical systems," *Acta Informatica*, vol. 43, no. 7, pp. 501–519, 2007.

[30] G. E. Bredon, *Topology and Geometry*. Springer, 1993.

[31] M. W. Hirsch, S. Smale, and R. L. Devaney, *Differential Equations, Dynamical Systems & An Introduction to Chaos*, 2nd ed. Elsevier, 2004.

Paper B

Complete Abstractions of Dynamical Systems by Timed Automata

Christoffer Sloth and Rafael Wisniewski

This paper was published in:
Nonlinear Analysis: Hybrid Systems, 2012

Abstract

This paper addresses the generation of complete abstractions of polynomial dynamical systems by timed automata. For the proposed abstraction, the state space is divided into cells by sublevel sets of functions. We identify a relation between these functions and their directional derivatives along the vector field, which allows the generation of a complete abstraction.

To compute the functions that define the subdivision of the state space in an algorithm, we formulate a sum of squares optimization problem. This optimization problem finds the best subdivisioning functions, with respect to the ability to approximate the dynamical system, in a subset of admissible subdivisioning functions.

1 Introduction

Formal verification is used to prove that a system satisfies a specification, e.g., "no system trajectories can reach an unsafe subset of the state space". Formal verification has been successfully developed in computer science for verifying various classes of models and specifications, e.g., to verify timed temporal logics of timed automata [1].

Similarly, formal verification methods have been developed for dynamical systems; see [2] for a survey. However, as the verification of system properties such as safety is based on reachability calculations, which in general are incomputable for continuous and hybrid systems [3], the developed methods are mostly approximate, and do not apply for general classes of systems. According to [2], the main source of complexity in the verification procedure of continuous and hybrid systems is the computation of the reachable set of the continuous dynamics.

Indirect verification methods are based on abstracting the considered systems by models of reduced complexity, while preserving certain properties of the original systems. This is accomplished for hybrid systems in [4], for continuous systems in [5], and for controller design in [6]. For relating the considered system with an abstraction of it, the notions of soundness and completeness are used. Roughly speaking, the reachable set of a sound (complete) abstraction includes (equals) the reachable set of the original system. Remark that the original system and the abstraction may be of different categories; hence, the relation between their reachable sets should be appropriately defined. This is accomplished in Section 3.

The class of direct verification methods does not abstract the dynamical system by another model, but handles the considered system directly. One such method is presented in [7], where the safety is determined based on the calculation of the backwards reachable set from a target set, by numerically calculating the solution of a Hamilton-Jacobi partial differential equation.

Generally, indirect methods allow expressive specifications, but simple dynamical models, and direct methods allow less expressive specifications, but more general system models. The method used in this paper is indirect, as it is based on abstracting the system by a timed automaton.

In this work, we strive to use the better of the two classes of methods. Hence, we will allow expressive specifications, but to obtain a close resemblance between the abstraction and a possibly complicated dynamical system, the abstraction is generated via a subdivision of the state space, which is conducted in accordance with the vector field.

We restrict the vector field to be a polynomial map, which allows algorithmic generation of the subdivision, similar to the algorithms presented in [8].

The proposed abstraction generates a timed automaton; hence, timed specifications can be considered, e.g., having a time-dependent unsafe set. The subdivisioning is inspired by [4] that uses foliations to divide the state space. Whereas, we use level sets of polynomial functions to generate invariant sets similar to [9]. The general framework of the abstraction method is presented in [10, 11]. In these works, no conditions for generating complete abstractions are provided; hence, this paper provides a step towards generating abstractions that preserves the reachability property.

The main contribution of this paper is to formulate a necessary and sufficient condition for the subdivision of the state space, under which a complete abstraction can be generated. This condition shows that the directional derivative of the subdivisioning function along the vector field is a function of the subdivisioning function itself. A second result is the development of an algorithm for computing the best subdivisioning function for a polynomial system.

Comparison to Related Work

The method used for abstracting models by timed automata in this paper has similarities with other verification and abstraction methods, but also has distinct properties, as e.g. the abstraction is a timed automaton in contrast to automata or transition systems used in most other works. In the following, we compare the method used in this paper (for short we call it TA-abstraction) to a few selected methods that are similar to TA-abstraction.

The principle of the barrier certificate method presented in [12] is to find a positive invariant set given as a sublevel set of a function (the level set is called a barrier) that includes the initial set, but excludes the unsafe set. If such a set can be found, the system is safe. Since only one function is used to separate the initial set and unsafe set, this function may be required to be very complicated. This is exemplified in the following.

Consider the two-dimensional linear system given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.1 & 1 \\ -1 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{7.1}$$

with initial set $X_0 = [-0.05,\, 0.05] \times [4.95,\, 5.05]$ and unsafe set $X_{\mathrm{u}} = [-0.05,\, 0.05] \times [0.85,\, 0.95]$.

To verify the safety of this system, a very high degree barrier certificate must be identified. A barrier certificate is illustrated in Figure 7.1 by the red line, which is the level set or barrier separating the initial set and unsafe set. Note that the barrier certificate is generated by hand, as an attempt to solve the problem in SOSTOOLS failed.

The barrier certificate is a special instance of the TA-abstraction, and this particular choice of subdivisioning function would result in an abstraction given by a timed automaton with two locations, where all solutions stay in the initial location for all times. However, the TA-abstraction allows the use of multiple subdivisioning functions; hence, these may be of lower degree at the cost of the need for several level sets.

Another related method is developed in [13, 14, 15], and is in the class of sign based abstractions described in [16]. The similarities between this method and the TA-abstraction is that level sets of multiple functions are used to subdivide the state space and that the transitions in the abstract model are generated based on the Lie derivative of

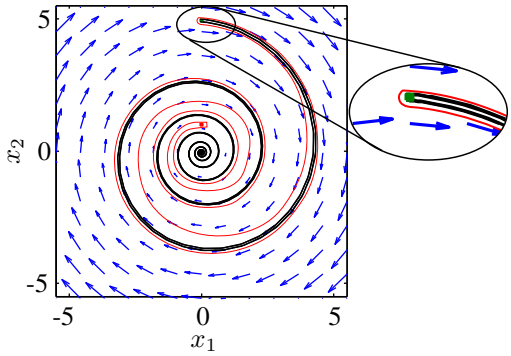


Figure 7.1: Illustration of the initial set (green box) and unsafe set (red box) together with some admissible system trajectories (block lines). The red line is the graph of a barrier certificate that proves the safety of the system.

the function used to subdivide the state space with respect to the vector field. In fact, the admissible subdivisioning functions in the TA-abstraction is only a subset of the admissible subdivisioning functions of this abstraction, since we require the Lie derivative of the subdivisioning functions with respect to the vector field to be decreasing (except at critical points). Additionally, we include time in the abstract model; hence, a more accurate calculation of the reachable set is obtained. This is clear from the following example.

Consider a dynamical system, given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{7.2}$$

The subdivision of the state space is illustrated in Figure 7.2 by black lines, and the set of initial states is shaded gray.

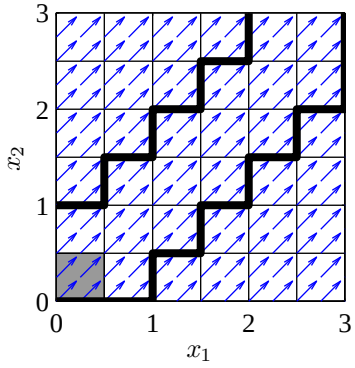


Figure 7.2: Phase plot of a system with constant derivatives. The system is initialized in the bottom left cell. If an automaton is generated based on the Lie derivative on the boundary of cells, we can only conclude that the reachable set is a subset of the first quadrant.

By use of the sign based abstraction, the entire first quadrant is reachable, but by use of TA-abstraction only a block diagonal trace is reachable (cells within the bold black line in Figure 7.2); see the full example in Section 7. Finally, we provide conditions for generating a complete abstraction, whereas this method only provides a heuristics for generating additional subdivisioning functions based on an initial collection of subdivisioning functions.

An abstraction method is presented in [17], which along the lines of the previously explained method, generates discrete abstractions of systems using the Lie derivative. As multi-affine systems (i.e., the degree of the vector field in any of the variables is less than or equal to 1) are considered on a rectangular subdivision of the state space, the computations of the Lie derivative can be done efficiently.

Methods for verifying systems based on the generation of positive invariant sets also exist. In [9], the generation of box shaped invariant sets is treated. This is motivated by examples from biology, but is not as general as the TA-abstraction as the shape of the invariant is constraint and the time is not used. Finally, in [18] control is used to shape the Lie derivative on the boundaries of rectangles to obtain some overall control objective.

When comparing TA-abstraction to the above mentioned abstractions, the complexity of generating the TA-abstraction seems to be higher. This is the case as time information should be generated in addition to the discrete structure of the sign based abstraction. However, as demonstrated in the examples, TA-abstraction improves the reachability calculations and allows the use of lower dimensional polynomials to generate the abstraction compared to the barrier certificate method.

This paper is organized as follows. Section 2 contains preliminary definitions, Section 3 explains how a dynamical system and an abstract model are related, and Section 4 explains how the state space is subdivided using level sets of functions. Section 5 describes how a timed automaton can be generated from the subdivision and gives a necessary and sufficient condition for generating a complete abstraction. Subsequently, Section 6 presents algorithms for synthesizing the abstraction, and an example is provided in Section 7. Finally, Section 8 comprises conclusions.

Notation

The set $\{1, \dots, k\}$ is denoted by $\boldsymbol{k}$. B^A is the set of maps $A \to B$. The power set of A is denoted 2^A. The cardinality of the set A is denoted $|A|$. We consider the Euclidean space $(\mathbb{R}^n, \langle , \rangle)$, where $\langle , \rangle$ is the standard scalar product. $\mathbb{N} = \{1, 2, \dots\}$ is the set of natural numbers, and $\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$ is the set of integers.

2 Preliminaries

The purpose of this section is to provide definitions related to dynamical systems and timed automata.

Dynamical Systems

A dynamical system $\Gamma = (X, f)$ has state space $X \subseteq \mathbb{R}^n$ and dynamics described by a system of ordinary differential equations $f : X \to \mathbb{R}^n$

$$\dot{x} = f(x). \tag{7.3}$$

Throughout the paper, we assume that f has non-degenerate critical points and that the vector field f is polynomial. This enables the use of algorithms in solving the problems.

We only consider polynomials with real-valued variables, and for $n \geq 1$ we denote the polynomial ring $\mathbb{R}[x_1, \ldots, x_n]$ by $\mathbb{R}[\underline{x}]$. In addition, a map $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be polynomial if its coordinate functions are polynomials, i.e., $f_i \in \mathbb{R}[\underline{x}]$ for $i = 1, \ldots, m$.

The solution of (7.3), from an initial state $x_0 \in X_0 \subseteq X$ at time $t \geq 0$ is described by the flow function $\phi_\Gamma : [0, \epsilon] \times X \to X$, $\epsilon > 0$ satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{7.4}$$

for all $t \in [0, \epsilon]$ and $\phi_\Gamma(0, x_0) = x_0$.

For a map $f : A \to B$, and a subset $C \subseteq A$, we define $f(C) \equiv \{f(x) |\, x \in C\}$. Thus, the reachable set is defined as follows.

Definition 76 (Reachable set of Dynamical System)**.** The reachable set of a dynamical system Γ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1, t_2]$ is

$$\phi_\Gamma([t_1, t_2], X_0). \tag{7.5}$$

Finally, we define a positive invariant set that is paramount for the subdivision of the state space.

Definition 77 (Positive Invariant Set)**.** Given a system $\Gamma = (X, f)$, a set $U \subseteq X$ is said to be positively invariant if for all $t \geq 0$

$$\phi_\Gamma(t, U) \subseteq U. \tag{7.6}$$

Timed Automata

We use the notation of [1] in the definition of a timed automaton. Let $\Psi(C)$ be a set of diagonal-free clock constraints for a set of clocks C. This set contains all invariants and guards of the timed automaton, and is described by the following grammar

$$\psi ::= c \bowtie k | \psi_1 \wedge \psi_2, \tag{7.7a}$$

where

$$c \in C,\ k \in \mathbb{R}_{\geq 0},\ \text{and}\ \bowtie \in \{\leq, <, =, >, \geq\}. \tag{7.7b}$$

Note that the clock constraint k should usually be a rational number, but in this paper, no effort is made to convert the clock constraints into rational numbers. However, any real number can be approximated by a rational number with an arbitrary small error $\epsilon > 0$. To make a clear distinction between syntax and semantics, the elements of $\bowtie$ are bold to indicate that they are syntactic operations. The semantics of the grammar is presented after the definition of a timed automaton.

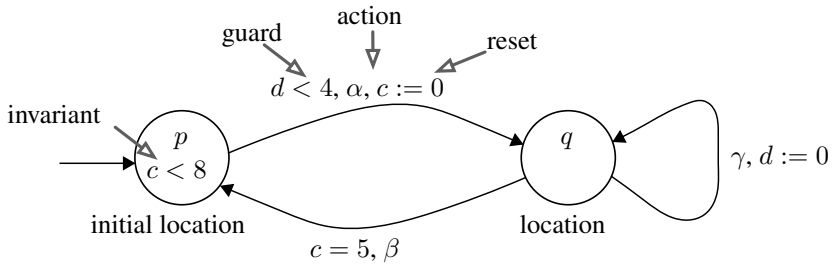


Figure 7.3: The transition between the location p and q with the label α may take place whenever the value of clock d is less than 4 and must take place before the value of clock c reaches 8. Once this transition occurs, the value of clock c is reset to 0.

Definition 78 (Timed Automaton)**.** A timed automaton $\mathcal{A}$ is a tuple $(E, E_0, C, \Sigma, I, \Delta)$, where

- E is a finite set of locations, and $E_0 \subseteq E$ is the set of initial locations;
- C is a finite set of clocks;
- Σ is the input alphabet;
- $I : E \to \Psi(C)$ assigns invariants to locations;
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations. A transition relation is a tuple $(e, G_{e\to e'}, \sigma, R_{e\to e'}, e')$ assigning an edge between two locations, where e is the source location and e' is the destination location. $G_{e\to e'} \in \Psi(C)$ is the set of guards, σ is a symbol in the alphabet Σ, and $R_{e\to e'} \subseteq C$ is a subset of clocks.

Example 13. A timed automaton is illustrated in Figure 7.3. The locations are denoted by p and q, where the initial location is p (indicated with a source less arrow); there are two clocks denoted by c and d, and actions designated by α, β, and γ. The transition between location p and q may take place whenever the value of clock d is less than 4, but must take place before the value of clock c reaches 8. The transition happens when the action α occurs. Once this transition is taken, the value of clock c resets to 0.

There is no invariant in q. This means that the system can be in location q no matter what the values of the clocks are.

The semantics of a timed automaton is defined in the following.

Definition 79 (Clock Valuation)**.** A clock valuation on a set of clocks C is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation v_0 is given by $v_0(c) = 0$ for all $c \in C$. For a valuation v, a scalar $d \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + d$ and $v[R]$ are defined as

$$(v + d)(c) = v(c) + d, \tag{7.8a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{7.8b}$$

It is seen that (7.8a) is used to progress time and (7.8b) is used to reset the clocks in the set R to zero.

We denote the set of maps $v : C \to \mathbb{R}_{\geq 0}$ by $\mathbb{R}_{\geq 0}^{C}$.

Remark 14: *This notation indicates that we identify a valuation v with C-tuples of nonnegative reals in $\mathbb{R}_{\geq 0}^{|C|}$, where $|C|$ is the number of elements in C. We impose the Euclidean topology on $\mathbb{R}_{\geq 0}^{C}$.*

Definition 80 (Semantics of Clock Constraint)**.** A clock constraint in $\Psi(C)$ is a set of clock valuations $\{v : C \to \mathbb{R}_{\geq 0}\}$ given by

$$[\![c \bowtie k]\!] = \{v : C \to \mathbb{R}_{\geq 0} | v(c) \bowtie k\} \tag{7.9a}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]. \tag{7.9b}$$

For convenience, we denote $v \in [\![\psi]\!]$ by $v \models \psi$ and denote the transition (e, v, σ, e', v') by $(e, v) \xrightarrow{\sigma} (e', v')$ in the following.

Definition 81 (Semantics of Timed Automaton)**.** The semantics of a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ is the transition system $[\![\mathcal{A}]\!] = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_{\mathrm{s}} \cup T_{\mathrm{d}})$, where S is the set of states

$$S = \{(e, v) \in E \times \mathbb{R}_{\geq 0}^{C} |\ v \models I(e)\},$$

$S_0 \subseteq S$ is the set of initial states

$$S_0 = \{(e, v) \in E_0 \times \mathbb{R}_{\geq 0}^{C} |\ v = v_0\}.$$

Note that $E \times \mathbb{R}_{\geq 0}^{C}$ induces subspace topology on S.

$T_{\mathrm{s}} \cup T_{\mathrm{d}}$ is the union of the following sets of transitions

$$\begin{aligned} T_{\mathrm{s}} = \{&(e, v) \xrightarrow{\sigma} (e', v') | \ \exists (e, G_{e \to e'}, \sigma, R_{e \to e'}, e') \in \Delta \\ &\text{such that } v \models G_{e \to e'} \text{ and } v' = v[R_{e \to e'}]\}, \\ T_{\mathrm{d}} = \{&(e, v) \xrightarrow{d} (e, v + d) | \ \forall d' \in [0, d] : v + d' \models I(e)\}. \end{aligned}$$

Hence, the semantics of a timed automaton is a transition system that comprises an infinite number of states: product of E and $\mathbb{R}_{\geq 0}^{C}$ and two types of transitions: the transition set T_{s} between discrete states with possibly a reset of clocks belonging to a subset $R_{e \to e'}$, and the transition set T_{d} that corresponds to time passing within the invariant $I(e)$.

In the following, we define an analog to the solution of a dynamical system for a timed automaton.

Definition 82 (Run of Timed Automaton)**.** A run of a timed automaton $\mathcal{A}$ is a possibly infinite sequence of alternations between time steps and discrete steps of the following form

$$(e_0, v_0) \xrightarrow{d_1} (e_0, v_1) \xrightarrow{\sigma_1} (e_1, v_2) \xrightarrow{d_2} \ldots, \tag{7.10}$$

where $d_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$.

By forcing alternation of time and discrete steps in Definition 82, the time step d_i is the maximal time step between the discrete steps σ_{i-1} and σ_i. The next example clarifies the semantics of a timed automaton.

Example 14. A timed automaton with two locations and two clocks is illustrated in Figure 7.3. All runs of the timed automaton start in the location p, and has initial valuation v_0. Furthermore, the time between an action α (a transition decorated by the label α) and an action β is 5 time units. There are infinitely many different runs of the timed automaton, and a few examples are

$$\begin{aligned}
&\text{Run 1}: (p,(0,0)) \xrightarrow{1} (p,(1,1)) \xrightarrow{\alpha} (q,(0,1)) \xrightarrow{2} (q,(2,3)) \xrightarrow{\gamma} \\
&(q,(2,0)) \xrightarrow{3} (q,(5,3)) \xrightarrow{\beta} (p,(5,3)) \to \dots \\
&\text{Run 2}: (p,(0,0)) \xrightarrow{3} (p,(3,3)) \xrightarrow{\alpha} (q,(0,3)) \xrightarrow{4} (q,(4,7)) \xrightarrow{\gamma} \\
&(q,(4,0)) \xrightarrow{1} (q,(5,1)) \xrightarrow{\beta} (p,(5,1)) \to \dots .
\end{aligned}$$

This finalizes the preliminary definitions. In the next section, we develop the necessary concepts for comparing solutions of a dynamical system with solutions of a timed automaton.

3 Abstractions of Dynamical Systems

To evaluate the generated abstraction, it is necessary to define a relation between solution trajectories of a dynamical system and runs of a timed automaton. This is accomplished in this section, by first defining the continuous behavior of a timed automaton, in terms of a trajectory. Second, we define a so-called abstraction function, which associates subsets of the state space to locations of a timed automaton. Finally, we define and construct sound and complete abstractions.

Trajectory of a Timed Automaton

A vital object for studying the behavior of any dynamical system is its trajectory. Therefore, we define a trajectory of a timed automaton [10]. At the outset, we introduce a concept of a time domain.

In the following, we denote sets of the form $\{a, \dots\}$ with $a \in \mathbb{Z}_{\geq 0}$ as $\{a, \dots, \infty\}$. Let $k \in \mathbb{N} \cup \{\infty\}$; a subset $\mathcal{T}_k \subset \mathbb{Z}_{\geq 0} \times \mathbb{R}_{\geq 0}$ with disjoint (union) topology will be called a time domain if there exists an increasing sequence $\{t_i\}_{i\in\{0,\dots,k\}}$ in $\mathbb{R}_{\geq 0} \cup \{\infty\}$ such that

$$\mathcal{T}_k = \bigcup_{i\in\{1,\dots,k\}} \{i\} \times T_i,$$

where

$$T_i = \begin{cases} [t_{i-1}, t_i] & \text{if } t_i < \infty \\ [t_{i-1}, \infty[& \text{if } t_i = \infty. \end{cases}$$

Note that $T_i = [t_{i-1}, t_i]$ for all i if $k = \infty$. We say that the time domain is infinite if $k = \infty$ or $t_k = \infty$. The sequence $\{t_i\}_{i\in\{0,\dots,k\}}$ corresponding to a time domain will be called a switching sequence.

We define two projections $\pi_1 : E \times \mathbb{R}^C_{\geq 0} \to E$ and $\pi_2 : E \times \mathbb{R}^C_{\geq 0} \to \mathbb{R}^C_{\geq 0}$ by $\pi_1(e, v) = e$ and $\pi_2(e, v) = v$.

Definition 83 (Trajectory)**.** A trajectory of the timed automaton $\mathcal{A}$ is a pair $(\mathcal{T}_k, \gamma)$ where $k \in \mathbb{N} \cup \{\infty\}$ is fixed, and

- $\mathcal{T}_k \subset \mathbb{Z}_{\geq 0} \times \mathbb{R}_{\geq 0}$ is a time domain with switching sequence $\{t_i\}_{i \in \{0,\dots,k\}}$,
- $\gamma : \mathcal{T}_k \to S$ and recall that S is the (topological) space of joint continuous and discrete states; see Definition 81 and Remark 14. The map γ satisfies:
  1. For each $i \in \{1, \dots, k-1\}$, there exists $\sigma \in \Sigma$ such that
  $$\gamma(i, t_i) \xrightarrow{\sigma} \gamma(i+1, t_i) \in T_{\mathrm{s}}.$$
  2. Let $\mathbf{0}$ be a vector of zeros and $\mathbf{1}$ be a vector of ones in $\mathbb{R}^C$. For each $i \in \{1, \dots, k\}$
  $$\pi_2(\gamma(i, t_{i-1} + d)) = \pi_2(\gamma(i, t_{i-1})) + d\mathbf{1} \;\; \forall d \in \begin{cases} [0,\, t_i - t_{i-1}] & \text{if } t_i < \infty \\ [0,\, \infty[& \text{if } t_i = \infty \end{cases}$$
  where $\pi_2(\gamma(i, t_{i-1} + d)) \in [\![I(\pi_1(\gamma(i, t_i)))]\!]$ and $\pi_2(\gamma(1, t_0)) = \mathbf{0}$. (Item 2 ensures that the time derivative of the valuation of each clock is one, between the discrete transitions.)

Note that γ is continuous by construction. Recall the definition of v_0 from Definition 79. A trajectory at (e, v_0) (with $v_0 \models I(e)$) is a trajectory $(\mathcal{T}_k, \gamma)$ with $\gamma(1, t_0) = (e, v_0)$.

We define a discrete counterpart of the flow map.

Definition 84 (Flow Map of Timed Automaton)**.** The flow map of a timed automaton $\mathcal{A}$ is a multivalued map
$$\phi_{\mathcal{A}} : \mathbb{R}_{\geq 0} \times S_0 \to 2^S,$$
defined by $(e', v') \in \phi_{\mathcal{A}}(t; e, v_0)$ if and only if there exists a trajectory $(\mathcal{T}_k, \gamma)$ at (e, v_0) such that $t = t_k - t_0$ and $(e', v') = \gamma(k, t_k)$.

It will be instrumental to define a discrete flow map $\Phi_{\mathcal{A}} : \mathbb{R}_{\geq 0} \times E_0 \to 2^E$, which forgets the valuation of the clocks
$$\Phi_{\mathcal{A}}(t, e) = \pi_1 \circ \phi_{\mathcal{A}}(t; e, v_0). \tag{7.11}$$

In other words, $\Phi_{\mathcal{A}}$ is defined by: $e' \in \Phi_{\mathcal{A}}(t, e)$ if and only if there exists a run (7.10) of $[\![\mathcal{A}]\!]$ initialized in (e, v_0) that reaches the location e' at time $t = \sum_i d_i$.

Example 15 (Continuation of Example 14)**.** In this example, the time domain $\mathcal{T}_k$ and trajectory of Run 1 in Example 14 are elucidated. The time domain is
$$\mathcal{T}_k = \{1\} \times [0, 1] \cup \{2\} \times [1, 3] \cup \{3\} \times [3, 6] \cup \{4\} \times [6, \dots.$$
From the time domain, it is seen that there are three discrete switches and that the total time of the run is 6 time units. The trajectory of the run is shown in Figure 7.4. To visualize the trajectory, the valuation of the clock c is illustrated by a blue solid line and the valuation of clock d is illustrated by a red dashed line. Furthermore, the current location of the automaton is indicated by its name.

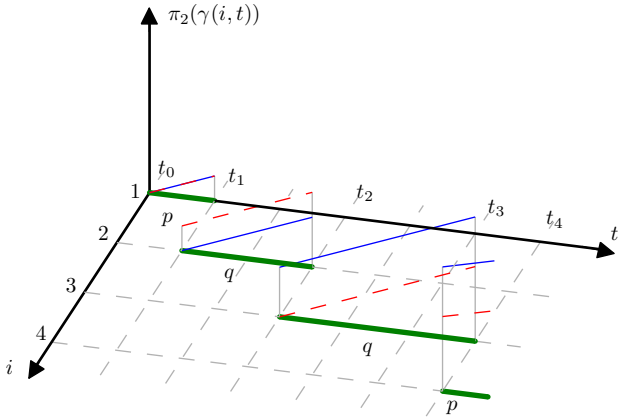


Figure 7.4: Trajectory of Run 1 in Example 14.

The reachable set of a timed automaton is defined as follows.

Definition 85 (Reachable set of Timed Automaton)**.** The reachable locations of a system $\mathcal{A}$ from a set of initial locations $E_0 \subseteq E$ on the time interval $[t_1, t_2]$ is defined as

$$\Phi_{\mathcal{A}}([t_1, t_2], E_0) \equiv \bigcup_{(t,e)\in[t_1,t_2]\times E_0} \Phi_{\mathcal{A}}(t, e). \tag{7.12}$$

Abstractions

We develop a concept of an abstraction of the dynamical system Γ. It consists of a finite number of sets $E \equiv \{e_\lambda |\ \lambda \in \Lambda\}$, called cells. The cells cover the state space X

$$X = \bigcup_{\lambda\in\Lambda} e_\lambda.$$

To the subdivision E, we associate an abstraction function, which to each point in the state space associates the cells that this point belongs to.

Definition 86 (Abstraction Function)**.** Let $E \equiv \{e_\lambda |\ \lambda \in \Lambda\}$ be a finite subdivision of the state space $X \subseteq \mathbb{R}^n$. An abstraction function for E is the multivalued function $\alpha_E : X \to 2^E$ defined by

$$\alpha_E(x) \equiv \{e \in E |\ x \in e\}. \tag{7.13}$$

For a given dynamical system Γ, we want to simultaneously devise a subdivision E of the state space X and create a timed automaton $\mathcal{A}$ with locations E such that

1. the abstraction is **sound** on an interval $[t_1, t_2]$:

$$\alpha_E \circ \Phi_\Gamma(t, X_0) \subseteq \Phi_{\mathcal{A}}(t, \alpha_E(X_0)), \text{ for all } t \in [t_1, t_2]$$

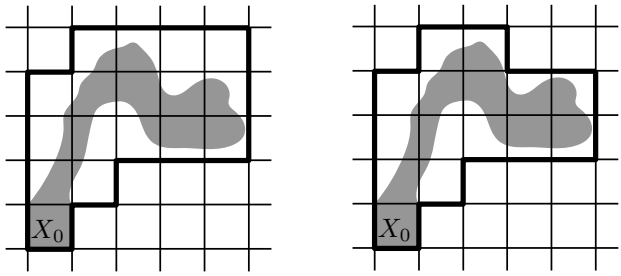


Figure 7.5: Reachable set of a dynamical system (shaded area), and reachable sets of automata (cells within bold lines). In the left figure, the reachable set of the automaton includes more cells than the ones reached by the dynamical system, i.e., the abstraction is sound. In the right figure, the reachable set of the automaton includes only the cells that are reached by the dynamical system, i.e., the abstraction is complete.

2. the abstraction is **complete** on an interval $[t_1, t_2]$:

$$\alpha_E \circ \Phi_\Gamma(t, X_0) = \Phi_\mathcal{A}(t, \alpha_E(X_0)) \text{ for all } t \in [t_1, t_2].$$

If a sound abstraction $\mathcal{A}$ is safe then Γ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$. Soundness is close to the notion of simulation; however, by soundness we relate different categories of models. Figure 7.5 illustrates the reachable set of a dynamical system, along with reachable sets of a sound abstraction (left) and a complete abstraction (right).

It is possible to describe the behavior of the original system by a transition system [19]; hence, allowing the generation of simulation and bisimulation relations between the original and abstract systems. This will not be pursued in details in this paper. However, if we define an observation map of the transition system describing the original system by associating states in cells with the cell, and add a discrete transition when a boundary of a cell is crossed, then a discrete abstraction simulates the original system and a complete abstraction is bisimilar to the original system.

In the remainder of the paper, we provide as answer to the following questions

- How should the state space be subdivided to allow the generation of a complete abstraction of a dynamical system?
- How can the subdivisioning functions be algorithmically found?

4 Subdividing the State Space

This section presents the method used for subdividing the state space by functions, but not how the functions should be chosen to obtain a complete abstraction. This is explained in Section 5. The subdivision of the state space is generated by intersecting sublevel sets of functions, and has two components: slices and cells. A slice is a sublevel set of one subdivisioning function, whereas a cell is a connected component of the intersection of sublevel sets of more functions.

The definition a subdivision is motivated by the definition of a complex in algebraic topology [20].

Definition 87. Let Λ be an index set, and $K = \{P_i\}_{i\in\Lambda}$ be a family of subsets in a Euclidean space $\mathbb{E}$. We let $|K| = \cup_{i\in\Lambda} P_i$ with the subspace topology inherited from $\mathbb{E}$. We call K a *subdivision* of a subset Y of $\mathbb{E}$, if

1. $\text{int}(P) \neq \emptyset$, for all $P \in K$,
2. $P \cap P^{'}$ belongs to the boundary of P and $P^{'}$ for all $P, P^{'} \in K$,
3. each point of $|K|$ has a neighborhood intersecting only finitely many elements of K,
4. $|K| = Y$.

Let $A \subseteq \mathbb{R}^n$, then $\text{cl}(A)$ denotes the closure of A. We define a slice as the set-difference of positively invariant sets.

Definition 88 (Slice)**.** A nonempty set S is a slice if there exist two open sets A_1 and A_2 such that

1. A_1 is a proper subset of A_2,
2. A_1 and A_2 are positively invariant (see Definition 77), and
3. $S = \text{cl}(A_2 \backslash A_1)$.

Since A_1 and A_2 are positively invariant sets, a trajectory initialized in S can propagate to A_1, but no solution initialized in A_1 can propagate to S. This implies that, via these invariants, we can study the possible trajectories of a dynamical system. We will adopt the convention that $\emptyset$ is a positively invariant set of any dynamical systems.

To devise a subdivision of a state space, we need to define collections of slices, called slice-families.

Definition 89 (Slice-Family)**.** Let $k \in \mathbb{N}$ and

$$A_0 \subset A_1 \subset \cdots \subset A_k$$

be a collection of positive invariant sets of a dynamical system $\Gamma = (X, f)$ with $X \subseteq A_k$ and $A_0 = \emptyset$. We say that the collection

$$\mathcal{S} \equiv \{S_i = \text{cl}(A_i \backslash A_{i-1}) |\ i \in \boldsymbol{k}\}$$

is a slice-family generated by the sets $\{A_i |\ i \in \boldsymbol{k}\}$ or just a slice-family.

We associate a function to each slice-family $\mathcal{S}$ to provide a simple way of describing the boundary of a slice. Such a function is called a subdivisioning function.

Definition 90 (Subdivisioning Function)**.** Let $\Gamma = (X, f)$ be a dynamical system and $\mathrm{Cr}(f)$ denote the set of critical points of f. Let $\mathcal{S}$ be a slice-family generated by the sets $\{A_i |\ i \in \boldsymbol{k}\}$. A continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \mathrm{Cr}(f)$ is a subdivisioning function for $\mathcal{S}$ if there is a sequence

$$a_0 < \ldots < a_k, \quad a_i \in \mathbb{R} \cup \{-\infty, \infty\},$$

where whenever $a_i \in \mathbb{R}$, it is a regular value of φ such that

$$\mathrm{cl}(A_i) = \varphi^{-1}([a_{i-1}, a_i]). \tag{7.14}$$

Remark that by regular level set theorem, for $a_i \in \mathbb{R}$, the boundary $\varphi^{-1}(a_i)$ of A_i is a smooth embedded submanifold of $\mathbb{R}^n$ of co-dimension 1 [21].

We will create cells that cover the entire state space, by intersecting slices. To ensure robustness of the subdivision, it is important that the slices intersect transversally. The robustness of a transversal intersection is readily seen from the definition of transversal intersection [22].

Definition 91 (Transversal Intersection)**.** Suppose that N_1 and N_2 are embedded submanifolds of M. We say that N_1 intersects N_2 transversally if, whenever $p \in N_1 \cap N_2$, we have $T_p(N_1) + T_p(N_2) = T_p(M)$. (The sum is not direct, just the set of sums of vectors, one from each of the two subspaces of the tangent space $T_p(M)$.)

The definition states that N_1 and N_2 are transversal if the tangent vectors to N_1 and N_2 span the entire space at each point of intersection. Hence, this transversality condition can be tested algorithmically.

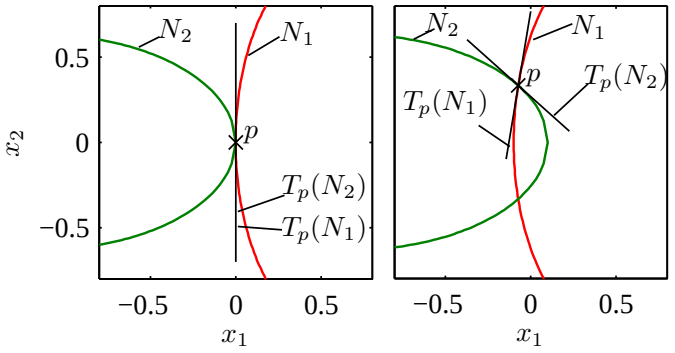


Figure 7.6: The left subplot shows an intersection that is not transversal; whereas, the right subplot shows a transversal intersection of two level sets.

The left subplot of Figure 7.6 illustrates level sets of two subdivisioning functions (hence two embedded submanifolds of $\mathbb{R}^2$). They intersect at the point p, and their tangents (black lines) generate a one dimensional subspace. This implies that their tangent vectors only span one dimension at p, i.e., $T_p(N_1) + T_p(N_2) \neq T_p(M)$. Therefore, this intersection is not transversal. Note that there exists an arbitrary small perturbation such that the intersection of the two level sets will be empty (this perturbation is given by a smooth map; see Theorem 2.1 in [22]). Therefore, this subdivision is not robust.

In the right subplot of Figure 7.6, two level sets intersecting at point p are illustrated. Their tangent vectors (black lines) span $\mathbb{R}^2$, i.e., the level sets intersect transversally. Note that two manifolds that do not intersect are also transversal.

We define transversal intersection of slices as follows.

Definition 92 (Transversal Intersection of Slices)**.** We say that the slices S_1 and S_2 intersect each other transversally and write

$$S_1 \pitchfork S_2 = S_1 \cap S_2 \tag{7.15}$$

if their boundaries, $\text{bd}(S_1)$ and $\text{bd}(S_2)$, intersect each other transversally.

Cells are generated via intersecting slices.

Definition 93 (Extended Cell)**.** Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of k slice-families and let

$$\mathcal{G}(\mathcal{S}) \equiv \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\} \subset \mathbb{N}^k.$$

Denote the j^{th} slice in $\mathcal{S}^i$ by S^i_j and let $g \in \mathcal{G}(\mathcal{S})$. Then

$$e_{\text{ex},g} \equiv \pitchfork_{i=1}^{k} S^i_{g_i}, \tag{7.16}$$

where g_i is the i^{th} component of the vector g. Any nonempty set $e_{\text{ex},g}$ is called an extended cell of $\mathcal{S}$.

The cells in (7.16) are denoted by extended cells, since the transversal intersection of slices may form multiple disjoint sets in the state space. It is desired to have cells, which are connected.

Definition 94 (Cell)**.** Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of k slice-families. A cell $e_{(g,h)}$ of $\mathcal{S}$ is a connected component of an extended cell of $\mathcal{S}$

$$\bigcup_h e_{(g,h)} = e_{\text{ex},g}, \text{ where} \tag{7.17a}$$

$$e_{(g,h)} \cap e_{(g,h')} = \emptyset \qquad \forall h \neq h'. \tag{7.17b}$$

There exist algorithms for determining the number of connected components of semi-algebraic sets given by polynomials (the number of connected components is given by the 0th Betti number [20]), and providing semi-algebraic descriptions of the connected components. One such algorithm is presented in [23]. Recall that $\mathbb{R}[x_1, \ldots, x_k]$ denotes the polynomial ring defined in Section 2. The algorithm takes as input a family of polynomials $\{P_1, \ldots, P_s\} \subset \mathbb{R}[x_1, \ldots, x_k]$ whose degrees are at most d, and outputs a semi-algebraic description of each connected component. The complexity of that algorithm is bounded by $s^{k+1}d^{O(k^3)}$, where O denotes the big-O notation [24, p. 252]. For more details on the algorithm, see chapter 15 in [25].

A finite subdivision based on the transversal intersection of slices is defined in the following.

Definition 95 (Finite Subdivision)**.** Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families. We define a finite subdivision $E(\mathcal{S})$ by

$$e \in E(\mathcal{S}) \tag{7.18}$$

if and only if e is a cell of $\mathcal{S}$.

We propose to use only functions that are decreasing along trajectories of the dynamical system Γ as subdivisioning functions, similar to Lyapunov functions, to obtain robustness of the subdivision. Indeed, the robustness is ensured as the vector field is transversal to the boundaries of the cells. This implies that there exists an arbitrary small perturbation of the vector field, such that it is still transversal to the boundary of the cells. The following definition origins from [26].

Definition 96 (Decreasing Subdivisioning Function)**.** Let X be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous, and recall that $\mathrm{Cr}(f)$ denotes the set of critical points of f. Then a real non-degenerate differentiable function $\varphi : X \to \mathbb{R}$ is said to be a subdivisioning function for f if

$$p \text{ is a critical point of } f \Leftrightarrow p \text{ is a critical point of } \varphi$$

$$L_f\varphi(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f^j(x) \tag{7.19a}$$

$$L_f\varphi(x) < 0 \ \ \forall x \in X \backslash \mathrm{Cr}(f) \tag{7.19b}$$

and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \mathrm{Cr}(f)$, where

$$||L_f\varphi(x)|| \geq \alpha ||x - p||^2. \tag{7.20}$$

We only require the vector field to be transversal to the level curves of a function φ, i.e., $L_f\varphi(x) = \langle \nabla\varphi(x), f(x) \rangle < 0$ for all $x \in X \backslash \mathrm{Cr}(f)$ and require nothing about the sign of φ.

In the next section, we present a procedure for generating a timed automaton from a subdivision, and show how the subdivisioning functions should be chosen to generate a complete abstraction.

5 Generation of Timed Automaton from Finite Subdivision

A timed automaton $\mathcal{A}$ is generated from a finite subdivision $E(\mathcal{S})$ as follows.

Definition 97 (Generation of Timed Automaton)**.** Given a finite collection of slice-families $\mathcal{S} = \{\mathcal{S}^i | \ i \in \boldsymbol{k}\}$, and $\mathcal{T} = \{(\underline{t}^i_{g_i}, \overline{t}^i_{g_i}) \in \mathbb{R}^2_{\geq 0} | \ i \in \boldsymbol{k}, \ g_i \in \{1, \ldots, |\mathcal{S}^i|\}\}$. The timed automaton $\mathcal{A}(\mathcal{S}, \mathcal{T}) = (E, E_0, C, \Sigma, I, \Delta)$ is defined by

- **Locations:** The locations of $\mathcal{A}$ are given by

$$E = E(\mathcal{S}). \tag{7.21}$$

  This means that a location $e_{(g,h)}$ is identified with the cell $e_{(g,h)} = \alpha^{-1}_{E(\mathcal{S})}(\{e_{(g,h)}\})$ of the subdivision $E(\mathcal{S})$, see Definition 86.

- **Clocks:** The set of clocks is $C = \{c^i | \ i \in \boldsymbol{k}\}$.

- **Alphabet:** The alphabet is $\Sigma = \{\sigma^i | \ i \in \boldsymbol{k}\}$.

- **Invariants:** In each location $e_{(g,h)}$, we impose an invariant

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \overline{t}^i_{g_i}. \tag{7.22}$$

- **Transition relations:** If a pair of locations $e_{(g,h)}$ and $e_{(g',h')}$ satisfy the following two conditions

  1. $e_{(g,h)}$ and $e_{(g',h')}$ are adjacent; that is $e_{(g,h)} \cap e_{(g',h')} \neq \emptyset$, and
  2. $g'_i \leq g_i$ for all $i \in \boldsymbol{k}$.

  Then there is a transition relation

$$\delta_{(g,h)\to(g',h')} = (e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma^i, R_{(g,h)\to(g',h')}, e_{(g',h')}),$$

  where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}^i_{g_i} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise.} \end{cases} \tag{7.23a}$$

  Note that $g_i - g'_i = 1$ whenever a transition labeled σ^i is taken.

  Let $i \in \boldsymbol{k}$. We define $R_{(g,h)\to(g',h')}$ by

$$c^i \in R_{(g,h)\to(g',h')} \tag{7.23b}$$

  iff $g_i - g'_i = 1$.

From the definition, it is seen that if the ith face is the common facet of the cells $e_{(g,h)}$ and $e_{(g',h')}$, then a transition is possible if the valuation of clock c^i is greater than $\underline{t}^i_{g_i}$. If the particular cells are not adjacent then no guard conditions are imposed.

If the set $\mathcal{S}$ is a singleton, i.e., $\mathcal{S} = \{\mathcal{S}_1\}$ then by slightly abusing the notation, we write $\mathcal{A}\left(\mathcal{S}_1, (\underline{t}, \overline{t})\right)$ instead of $\mathcal{A}\left(\{\mathcal{S}_1\}, \{(\underline{t}, \overline{t})\}\right)$.

We are interested in verifying a temporal logic statement over some regions of the state space; however, as the locations of the abstraction correspond to cells of the subdivision of the state space, level sets must be chosen such that the regions defined by predicates in temporal logic are subsets of a union of cells of the subdivision. Finally, the actual verification is conducted on the over-approximation of the regions involved in the temporal logic statement.

To ensure that the properties of an abstraction is not only valid for a particular choice of level sets, we impose the condition for any choice of regular values in the subdivision.

From Definition 97, it is seen that to generate a timed automaton, it is required to devise a subdivision of the state space, and a set of invariant and guard conditions. Therefore, we provide condition under which an abstraction is complete, recall the definition of a complete abstraction in Section 3.

Proposition 34: *Given a dynamical system $\Gamma = (X, f)$, a collection of subdivisioning functions $\{\varphi^i |\ i \in \boldsymbol{k}\}$, a collection of regular values $\{a^i_{g_i} |\ i \in \boldsymbol{k},\ g_i \in \{1, \ldots, |\mathcal{S}^i|\}\}$ generating $\mathcal{S}$, and $\mathcal{T} = \{(\underline{t}^i_{g_i}, \overline{t}^i_{g_i}) |\ i \in \boldsymbol{k},\ g_i \in \{1, \ldots, |\mathcal{S}^i|\}\}$. The timed automaton $\mathcal{A}(\mathcal{S}, \mathcal{T})$ is a complete abstraction of Γ if and only if for any $i \in \boldsymbol{k}$*

1. *for any $g \in \mathcal{G}(\mathcal{S})$ (see the definition of $\mathcal{G}(\mathcal{S})$ in Definition 93), such that $g_i \geq 2$ there exists a time $t^i_{g_i}$ such that for all $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$*

$$\phi_\Gamma(t^i_{g_i}, x_0) \in (\varphi^i)^{-1}(a^i_{g_i-1}) \tag{7.24}$$

and

2. $\bar{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$.

The proof of Proposition 34 can be found in the Appendix A.

Proposition 34 does not provide a straightforward method for computing a complete abstraction, as the conditions are not numerically tractable. Therefore, we rephrase (7.24) as a relation between the level sets of the subdivisioning function and its derivative along the vector field. This is given in the following theorem.

Theorem 12*: Let $\Gamma = (X, f)$ be a dynamical system. There exists a complete abstraction of Γ if and only if there exists a collection of subdivisioning functions $\{\varphi^i |\ i \in \boldsymbol{k}\}$, such that for any φ and any regular value $a \in \mathbb{R}$ there exists $b \in \mathbb{R}$ such that*

$$\{x \in \mathbb{R}^n |\ \varphi(x) - a = 0\} \subseteq \{x \in \mathbb{R}^n |\ L_f\varphi(x) - b = 0\}, \tag{7.25}$$

Proof. We show that (7.24) implies (7.25). Let $x'_0, x''_0 \in (\varphi)^{-1}(a)$ by (7.24)

$$\varphi(\phi_\Gamma(t, x'_0)) = \varphi(\phi_\Gamma(t, x''_0)) \qquad \forall t \in \mathbb{R}. \tag{7.26}$$

We differentiate with respect to t

$$\sum_i \frac{\partial \varphi}{\partial x_i}(\phi_\Gamma(t, x'_0)) f_i(\phi_\Gamma(t, x'_0)) = \sum_i \frac{\partial \varphi}{\partial x_i}(\phi_\Gamma(t, x''_0)) f_i(\phi_\Gamma(t, x''_0)) \qquad \forall t \in \mathbb{R}. \tag{7.27a}$$

$$L_f\varphi(\phi_\Gamma(t, x'_0)) = L_f\varphi(\phi_\Gamma(t, x''_0)) \qquad \forall t \in \mathbb{R}. \tag{7.27b}$$

Let $t = 0$, then $L_f\varphi(x'_0) = L_f\varphi(x''_0)$. Hence, (7.25) is satisfied.

To show that (7.25) implies (7.24), we define a convenient state transformation inspired by [27, p. 13]. In the new coordinates, the vector field has only one nonzero component.

Let $M = \varphi^{-1}(a)$ be a smooth manifold. Define the smooth function $\eta : M \to \mathbb{R}$ as

$$\eta(x) \equiv \frac{1}{||\nabla\varphi||^2}(x), \tag{7.28}$$

where $\nabla\varphi$ is the gradient of φ. Define the vector field ξ as

$$\dot{x} = \eta(x)\nabla\varphi(x) = \xi(x). \tag{7.29}$$

Let $\langle \nabla\xi, f\rangle \equiv D\xi(f)$ for all f. Then the total derivative of $\varphi(\xi)$ is

$$D\varphi(\xi) = \langle \nabla\varphi, \xi\rangle = \eta\langle \nabla\varphi, \nabla\varphi\rangle = 1, \tag{7.30}$$

Now, we define the map $F(\cdot,\cdot) : M \times [a, b] \to \varphi^{-1}([a, b])$ as

$$t \mapsto \varphi \circ F(x_0, t) = c + t, \tag{7.31}$$

where $F(x_0, t)$ is the solution of ξ from initial state x_0. Now we use new coordinates $x = F(q)$; hence,

$$\dot{x} = DF(q)\dot{q} = f(x) \tag{7.32a}$$

$$\dot{q} = \underbrace{(DF(q))^{-1} f(F(q))}_{=\tilde{f}(q)} \tag{7.32b}$$

We rewrite φ in q-coordinates

$$\varphi(x) = \underbrace{\varphi \circ F(q)}_{=\tilde{\varphi}(q)} \tag{7.33}$$

Let $x_1, x_2 \in M$, then by (7.25)

$$D\varphi f(x_1) = D\varphi f(x_2). \tag{7.34}$$

From (7.33), we get

$$d\tilde{\varphi}(q) = d\varphi(F(q))D(F(q))(q) \tag{7.35}$$

and in $\tilde{f}(q)$

$$d\tilde{\varphi}_q(\tilde{f}(q)) = d\varphi_{F(q)} D(F(q))_q (DF(q))^{-1} f(F(q)) \tag{7.36}$$

$$= d\varphi_{F(q)} f(F(q)). \tag{7.37}$$

It is seen that $\tilde{\varphi}$ only depends on the last coordinate. Therefore,

$$\tilde{\psi} = D\tilde{\varphi}\tilde{f} = \tilde{f}_n. \tag{7.38}$$

Since $\psi(x_1, t) = \psi(x_2, t)$ for any pair $x_1, x_2 \in M$ and $t \in [a, b]$, we have $\tilde{f}_n(x_1, t) = \tilde{f}_n(x_2, t)$. In other words, the nth component of the vector field $\tilde{f}$ is independent of its first $n - 1$ coordinates. As a consequence the inclusion (7.24) holds. □

Corollary 7: *From* (7.26) *in the proof of Theorem 12, the time* $\bar{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$ *in Proposition 34 can be calculated for a slice* $S^i_{g_i} = (\varphi^i)^{-1}([a, b])$ *by simulating a trajectory from any point* $x_0 \in (\varphi^i)^{-1}(b)$ *and determining the time* $t^i_{g_i}$, *when* $\phi_\Gamma(t^i_{g_i}, x_0) \in (\varphi^i)^{-1}(a)$.

In the remainder of the section, we identify the necessary relation between polynomials $\varphi, \psi \in \mathbb{R}[\underline{x}]$ that satisfy (7.25). Definitions related to polynomials are provided in A.

In this work, we are only interested in using irreducible polynomials to subdivide the state space, as they have a closed and connected variety [28, p. 90]. By using irreducible polynomials, the generated slices are connected components, as desired. Finally, we provide the relationship between polynomials $\varphi, \psi \in \mathbb{R}[\underline{x}]$ that satisfy (7.25).

Theorem 13: *Let* $\varphi, \psi \in \mathbb{R}[\underline{x}]$. *For any regular value* $a \in \mathbb{R}$, *there exists* $b \in \mathbb{R}$ *such that*

$$\{x \in \mathbb{R}^n |\ \varphi(x) - a = 0\} \subseteq \{x \in \mathbb{R}^n |\ \psi(x) - b = 0\}, \tag{7.39}$$

where $\varphi - a$ *is irreducible if and only if*

$$\psi = c_0 + c_1\varphi + c_2\varphi^2 + c_3\varphi^3 + \cdots. \tag{7.40}$$

Proof. First, we show that (7.40) implies (7.39). Let $\tilde{x} \in \{x \in \mathbb{R}^n | \, \varphi(x) - a = 0\}$ then by (7.40), $\psi(\tilde{x})$ is constant, i.e., there exists $b \in \mathbb{R}$ such that (7.39) is true.

The proof of the converse is more involved. By Theorem 14 from [29, p. 178], condition (7.39) is equivalent to the existence of $b \in \mathbb{R}$ for any regular value $a \in \mathbb{R}$ such that

$$(\psi - b) \in I(Z(\varphi - a)).$$

Theorem 14 states that for irreducible polynomials

$$I(\varphi - a) = I(Z(\varphi - a)).$$

Finally, by definition 101, this makes (7.39) equivalent to the existence of $p \in \mathbb{R}[\underline{x}]$ and $b \in \mathbb{R}$ for any regular value $a \in \mathbb{R}$ such that

$$\psi(x) - b = p(x)(\varphi(x) - a). \tag{7.41}$$

Now we show that (7.41) is indeed equivalent to $\psi(x)$ and $\varphi(x)$ being related as shown in (7.40). Pick $a, a', a'' \in \mathbb{R}$ any three regular values of $\varphi(x)$ (such a choice is possible by Sard's theorem, as the set of critical values of φ has measure zero [30].), $b, b', b'' \in \mathbb{R}$, and let $p, p', p'' \in \mathbb{R}[\underline{x}]$ such that

$$\psi(x) - b = p(x)(\varphi(x) - a) \tag{7.42}$$
$$\psi(x) - b' = p'(x)(\varphi(x) - a') \tag{7.43}$$
$$\psi(x) - b'' = p''(x)(\varphi(x) - a'') \tag{7.44}$$

For any $\tilde{x} \in \varphi^{-1}(a')$ then (7.42) becomes

$$b' - b = p(\tilde{x})(a' - a) \tag{7.45}$$
$$p(\tilde{x}) = \frac{b' - b}{a' - a} = \alpha. \tag{7.46}$$

It follows that

$$\{x \in \mathbb{R}^n | \, \varphi(x) - a' = 0\} \subseteq \{x \in \mathbb{R}^n | \, p(x) - \alpha = 0\} \tag{7.47}$$

By (7.41), this is equivalent to

$$p(x) - \alpha = q(x)(\varphi(x) - a'). \tag{7.48}$$

This implies that (7.42) is equivalent to the existence of $q \in \mathbb{R}[x_1, \ldots, x_n]$ such that

$$\psi(x) = b + (\alpha + q(x)(\varphi(x) - a'))(\varphi(x) - a). \tag{7.49}$$

By inserting $\tilde{x}$ into (7.44) and repeating the previous procedure, we see that (7.44) is equivalent to the existence of $q'' \in \mathbb{R}[\underline{x}]$ such that

$$\psi(x) = b'' + (\alpha'' + q''(x)(\varphi(x) - a'))(\varphi(x) - a''), \tag{7.50}$$

where $\alpha'' = (b' - b'')/(a' - a'')$. It is seen that (7.49) is equal to (7.50). Now let $\bar{x} \in \varphi^{-1}(a'')$, then

$$b + (\alpha + q(\bar{x})(a'' - a'))(a'' - a) = b'' \tag{7.51}$$

$$q(\bar{x}) = \frac{\frac{b''-b}{a''-a} - \alpha}{a'' - a'} = \bar{\alpha}. \tag{7.52}$$

By rewriting $q(x)$, similar to $p(x)$ previously, (7.49) is equivalent to

$$\psi(x) = b + (\alpha + (\bar{\alpha} + r(x)(\varphi(x) - a''))(\varphi(x) - a'))(\varphi(x) - a). \tag{7.53}$$

By continuing this procedure, we see that $\psi(x)$ should be some polynomial in $\varphi(x)$, as shown in (7.40). By hypothesis, ψ has a finite degree; hence, the procedure ends. □

Essentially, (7.40) states that ψ must be a polynomial in φ to ensure that any level set of φ is a subset of some level set of ψ.

6 Computation of Subdivisioning Functions

The purpose of this section is to show how to find subdivisioning functions that generate a complete abstraction. The functions are found by solving either a linear or a sum of squares optimization problem. If functions that allow the generation of a complete abstraction cannot be found with the imposed structure, the optimization problems provide the best functions for generating a sound abstraction. In this context, a good choice of subdivisioning functions gives a small difference between the times used in the guard and invariant conditions.

Problem Formulation

We cannot search for polynomials φ and ψ with the relation shown in (7.40) using linear or sum of squares optimization problems, as the coefficients of the polynomials would appear in different powers. Therefore, we assume that ψ is an affine function of φ.

Proposition 35*: Given a dynamical system* $\Gamma = (X, f)$, $\underline{c}_0, \underline{c}_1, \overline{c}_0, \overline{c}_1 \in \mathbb{R}$, *and a subdivisioning function* φ *such that*

$$\underline{c}_0 + \underline{c}_1\varphi(x) \leq L_f\varphi(x) \leq \overline{c}_0 + \overline{c}_1\varphi(x) \qquad \forall x. \tag{7.54}$$

Let $\mathrm{CrV}(\varphi)$ *be the set of critical values of* φ*. Then for any pair of regular values* $a_1 < a_2$*, where* $\mathrm{CrV}(\varphi) \cap [a_1,\, a_2] = \emptyset$*, and any* $x_0 \in \varphi^{-1}(a_2)$ *there exists* $t > 0$ *that satisfies*

$$\mathrm{sign}(\underline{c}_1 a_2 + \underline{c}_0)t \leq \mathrm{sign}(\underline{c}_1 a_2 + \underline{c}_0)\frac{1}{\underline{c}_1}\log(\frac{\underline{c}_0}{\underline{c}_1})\log(\frac{a_2}{a_1}),\ \textit{and} \tag{7.55a}$$

$$\mathrm{sign}(\overline{c}_1 a_2 + \overline{c}_0)t \geq \mathrm{sign}(\overline{c}_1 a_2 + \overline{c}_0)\frac{1}{\overline{c}_1}\log(\frac{\overline{c}_0}{\overline{c}_1})\log(\frac{a_2}{a_1}) \tag{7.55b}$$

such that

$$\phi_\Gamma(t, x_0) \in \varphi^{-1}(a_1). \tag{7.56}$$

Proof. A subdivisioning function is defined to be decreasing for all $x \in \mathbb{R}^n \backslash \mathrm{Cr}(f)$; hence, φ is decreasing for all $x \in \varphi^{-1}(a)$, where a is a regular value, i.e., $a \neq \mathrm{CrV}(\varphi)$. Therefore, any solution initialized in $\varphi^{-1}(a_2)$ reaches $\varphi^{-1}(a_1)$ in some finite time. If $\psi(x) = \overline{c}_0 + \overline{c}_1\varphi(x)$, we have

$$\varphi(t) = e^{\overline{c}_1 t}\varphi(0) + \int_0^t e^{(t-s)\overline{c}_1}\overline{c}_0 ds \tag{7.57}$$

$$\varphi(t) = e^{\overline{c}_1 t}\varphi(0) + \frac{\overline{c}_0}{\overline{c}_1}(e^{\overline{c}_1 t} - 1). \tag{7.58}$$

For some $\varphi(t)$ and $\varphi(0)$, it is seen that

$$t = \frac{1}{\overline{c}_1}\log(\frac{\underline{c}_0}{\overline{c}_1})\log(\frac{\varphi(t)}{\varphi(0)}). \tag{7.59}$$

Substituting this in the inequality, with $\varphi(0) = a_2$ and $\varphi(t) = a_1$

$$\mathrm{sign}(\overline{c}_1 a_1 + \overline{c}_0)t \leq \mathrm{sign}(\overline{c}_1 a_1 + \overline{c}_0)\frac{1}{\overline{c}_1}\log(\frac{\overline{c}_0}{\overline{c}_1})\log(\frac{a_1}{a_2}). \tag{7.60}$$

The second condition can be calculated in a similar manner. □

Note that $\psi(x)$ is required to be negative for any $x \in X \backslash \mathrm{Cr}(f)$. Therefore, $(\overline{c}_1 a_1 + \overline{c}_0)$ and $(\underline{c}_1 a_1 + \underline{c}_0)$ should also be negative; hence, (7.55a) gives a lower bound on t and (7.55b) gives an upper bound on t. Furthermore, it is seen from (7.55) that for $\underline{c}_0 = \overline{c}_0$ and $\underline{c}_1 = \overline{c}_1$, the abstraction generated by $\varphi(x)$ is complete. Otherwise, we minimize the time interval given by (7.55), to obtain the most accurate sound abstraction. This is accomplished by minimizing the upper bound and maximizing the lower bound on ψ given in (7.54). The procedure for this is provided next.

First, we make the polynomials φ, ψ homogeneous, to eliminate the constants $\underline{c}_0$ and $\overline{c}_0$.

Definition 98 ([29])**.** A homogeneous polynomial (or form) is a polynomial where all monomials have the same total degree d.

We can transform a polynomial into a homogeneous polynomial as follows.

Lemma 7 ([29])**.** *Let f be any polynomial in $\mathbb{R}[\underline{x}]$ of degree less than or equal to d. Then*

$$\bar{f}(x_0, \ldots, x_n) = x_0^d f(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}) \tag{7.61}$$

is a homogeneous polynomial of degree d.

It is very important to choose the degree d correctly, when generating the homogeneous polynomial, to ensure the following property.

Lemma 8. *Let f and $\bar{f}$ be related by* (7.61)*. If d is even, then $f(x) \geq 0$ for all $x \in \mathbb{R}^n$ if and only if $\bar{f}(x) \geq 0$ for all $x \in \mathbb{R}^{n+1}$.*

Proposition 36: *Let* $\varphi, \psi \in \mathbb{R}[\underline{x}]$ *and* $c_0, c_1 \in \mathbb{R}$. *Then*

$$\psi(x) \leq c_0 + c_1\varphi(x) \quad \forall x \in \mathbb{R}^n \tag{7.62}$$

if and only if

$$\bar{\psi}(y) \leq 1 + c_1\bar{\varphi}(y) \quad \forall y \in \mathbb{R}^{n+1}, \tag{7.63}$$

where $\bar{\varphi}, \bar{\psi} \in \mathbb{R}[x_0, \ldots, x_n]$ *are homogeneous polynomials of* φ, ψ.

Proof. Let $\varphi, \psi \in \mathbb{R}[\underline{x}]$ and $c_0, c_1 \in \mathbb{R}$ and let

$$\psi(x) \leq c_0 + c_1\varphi(x) \quad \forall x \in \mathbb{R}^n. \tag{7.64}$$

Then

$$\psi(x\sqrt[d]{c_0}) \leq c_0 + c_1\varphi(x\sqrt[d]{c_0}) \quad \forall x \in \mathbb{R}^n. \tag{7.65}$$

Let $\bar{\varphi}, \bar{\psi} \in \mathbb{R}[x_0, \ldots, x_n]$ be homogeneous polynomials of φ, ψ, then by definition of a homogeneous polynomial

$$c_0\bar{\psi}(y) \leq c_0 + c_0c_1\bar{\varphi}(y) \quad \forall y \in \mathbb{R}^{n+1}, \tag{7.66}$$

hence

$$\bar{\psi}(y) \leq 1 + c_1\bar{\varphi}(y) \quad \forall y \in \mathbb{R}^{n+1}. \tag{7.67}$$

□

From Proposition 36, we see that by considering homogeneous polynomials, only one decision variable is needed to obtain each bound of (7.54).

Generation of Optimization Problem

As the vector field and the subdivisioning functions are polynomial, we will use sum of squares optimization problems to compute subdivisioning functions. The following explanation of sum of squares polynomials is based on [31, 32].

Definition 99. A polynomial $p \in \mathcal{P}_{n,d}$ is called sum of squares (SOS) if

$$p = \sum_{i=1}^{k} p_i^2 \tag{7.68}$$

for some polynomials $p_i \in \mathcal{P}_n$ with $i = 1, \ldots, k$.

The set of sum of squares polynomials is a subset of nonnegative polynomials [31], which can be treated using semidefinite programming, as described below. We denote the set of sum of squares polynomials in n variables by Σ_n.

The existence of a sum of squares decomposition of a polynomial $p \in \mathcal{P}_{n,d}$ can be expressed as a semidefinite programming feasibility problem. Therefore, the formulation

of a problem as sum of squares makes the problem computationally tractable; however, the number of decision variables in the program is

$$N = \binom{n+2d}{2d}. \tag{7.69}$$

In the search of sum of squares polynomials, it is exploited that the existence of an SOS decomposition of a polynomial p is equivalent to the existence of a matrix $Q = Q^{\mathrm{T}} \geq 0$ such that

$$p = Z^{\mathrm{T}} Q Z, \tag{7.70}$$

where Z is a vector of monomials of degree less than or equal to half the degree of p.

Let $k, l \in \mathbb{Z}_{>0}$, let $\alpha_{i,j} \in \mathbb{R}[\underline{x}]$ for $(i,j) \in k \times l$, and $w_j \in \mathbb{R}$. An SOS programming problem is

$$\begin{aligned} &\underset{(c_1,\ldots,c_l)\in\mathbb{R}^l}{\text{minimize}} \sum_{j=1}^{l} w_j c_j \text{ subject to} \\ &\alpha_{i,0} + \sum_{j=1}^{l} \alpha_{i,j} c_j \in \Sigma_n \forall i = 1, \ldots, k. \end{aligned} \tag{7.71}$$

It is seen that an SOS programming problem is a minimization of a linear cost subject to SOS feasibility constraints.

The main issue with sum of squares polynomials is that a polynomial may be nonnegative, even though it cannot be represented as a sum of squares [32].

Now, we can find the subdivisioning function that gives the best abstraction, via a sum of squares optimization problem. Note that this is very similar to the problem of finding the maximum decay rate of a system, which can be formulated as a generalized eigenvalue problem. This problem can be solved by the bisection method.

In the considered problem, we do not assume the system to be stable nor unstable; hence, we cannot assume any sign of the decay rate. However, to avoid complicating the optimization problems, we assume that the decay rate is positive (in the optimization problem $\gamma > 0$), but if $\gamma < 0$ all maximizations should just be replaced with minimizations and vice versa.

Proposition 37*: Suppose $\Gamma = (X, f)$ is a polynomial system, then the subdivisioning function φ, minimizing the upper bound of* (7.63) *is given by the following optimization problem*

$$\begin{aligned} &\max \gamma \text{ subject to} \\ 1 + \gamma\varphi - L_f\varphi &\in \Sigma_n \\ -L_f\varphi &\in \Sigma_n \\ \gamma &> 0 \end{aligned} \tag{7.72}$$

and the subdivisioning function φ, maximizing the lower bound of (7.63) *is given by the following optimization problem*

$$\begin{aligned} &\min \gamma \text{ subject to} \\ -1 - \gamma\varphi + L_f\varphi &\in \Sigma_n \\ -L_f\varphi &\in \Sigma_n \\ \gamma &> 0. \end{aligned} \tag{7.73}$$

These optimization problems can be solved using the bisection method [33]. Note that the previous optimization problems can be solved in tools such as SOSTOOLS [34]. This is a tool that transforms sum of squares optimization problems into semidefinite programs (SDPs), which can be directly solved using, e.g., SeDuMi. Solving SDPs is polynomial time [35]; hence, the sum of squares optimization problem can be solved in polynomial time, with the number of decision variables given by (7.69).

To clarify the statement, we also pose the optimization problem using LMIs, which can be used for linear systems with quadratic subdivisioning functions. To say that a symmetric matrix M is positive definite (positive semidefinite), we write $M \succ 0$ ($M \succeq 0$).

Corollary 8: *Suppose* $\Gamma = (X, f)$ *is a linear system, with* $f(x) = Ax$, *and let* $\varphi(x) = x^{\mathrm{T}}Px$, *where* P *is a symmetric* $n \times n$ *matrix. Define the matrix* Q *as*

$$Q \equiv A^{\mathrm{T}}P + PA. \tag{7.74}$$

Then the upper bound of (7.63) *is given by the following optimization problem*

$$\begin{aligned} &\textit{max } \gamma \textit{ subject to} \\ &\gamma P - Q \succeq 0 \\ &Q \prec 0, \gamma > 0. \end{aligned} \tag{7.75}$$

and the subdivisioning function φ, *maximizing the lower bound of* (7.63) *is given by the following optimization problem*

$$\begin{aligned} &\textit{min } \gamma \textit{ subject to} \\ &\gamma P - Q \preceq 0 \\ &Q \prec 0, \gamma > 0. \end{aligned} \tag{7.76}$$

To show that functions generating complete abstractions do exist, we provide subdivisioning functions for a linear and a polynomial system in the following.

Consider the following linear system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1.275 & 1.25 \\ -2.3125 & -1.475 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{7.77}$$

The subdivisioning function

$$\varphi(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 74 & 44 \\ 44 & 40 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7.78}$$

satisfies the condition for completeness, as $\varphi = 5L_f\varphi$.

For the polynomial system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_1^3 x_2 \\ x_1^2 x_2^2 \end{bmatrix}, \tag{7.79}$$

a subdivisioning function

$$\varphi(x) = x_1^2 x_2 \tag{7.80}$$

is related to its derivative along the vector field via the relation $3\varphi^2 = L_f\varphi$.

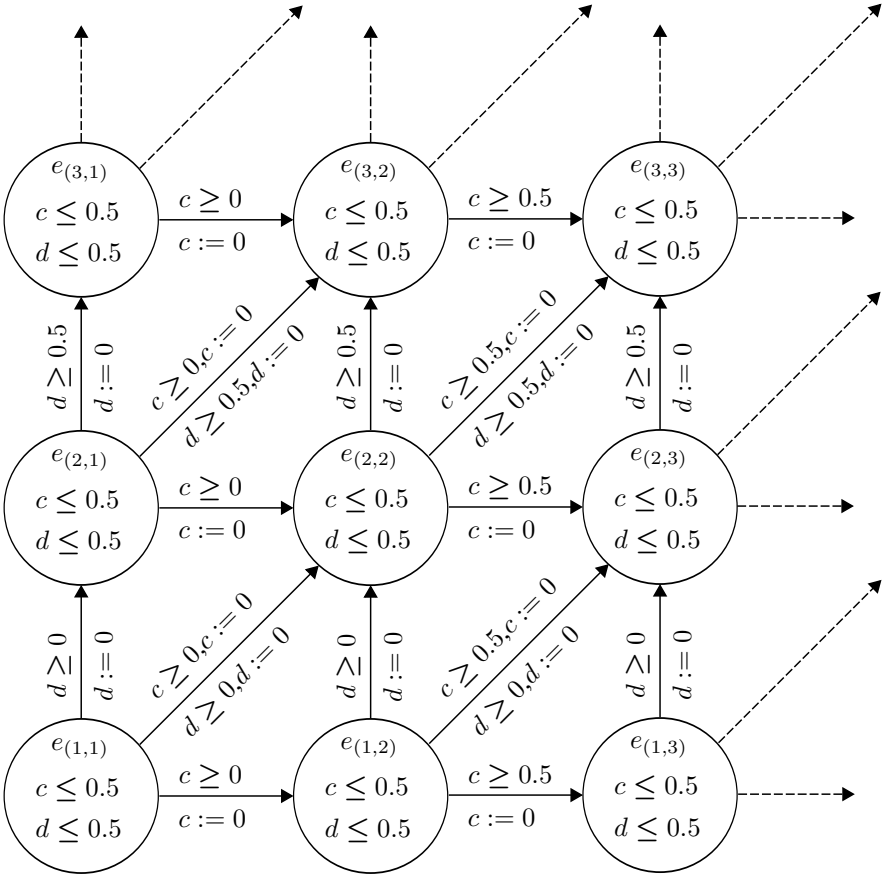


Figure 7.7: Timed automaton for the abstraction of the example at Figure 7.2.

7 Illustrative Example

In this section, we provide an abstraction of the system introduced in Section 1 and an illustrative example of a subdivision of a state space for a polynomial dynamical system.

For example of a subdivision of the state space shown at Figure 7.2, the TA-abstraction generates a timed automaton shown in Figure 7.7, where only the first nine locations are illustrated.

The abstraction is complete; hence, it takes the same time for any solution to traverse a slice. For the particular subdivision, where all cells are squares with a width and height of 0.5, it takes 0.5 time units for any solutions to traverse a slice; hence, all guards and invariants are 0.5 time units.

In the next example, we consider a two dimensional system that has the following polynomial vector field

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2 \\ x_1 - x_2 - 2 \end{bmatrix} \tag{7.81}$$

and has a stable critical point in $(x_1, x_2) = (-2, -4)$ and a saddle point in $(x_1, x_2) = (1, -1)$. To give an intuition about the behavior of trajectories of the vector field, the vector field and nullclines are shown in Figure 7.8.

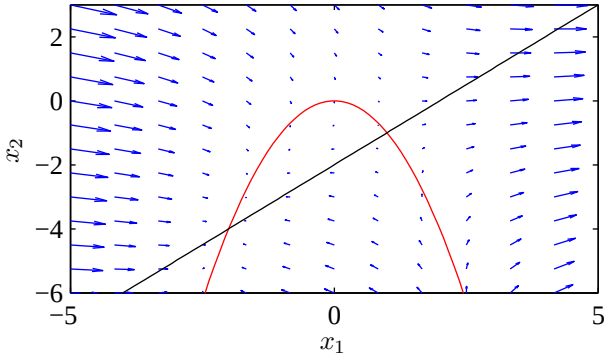


Figure 7.8: Vector field with two equilibria. The two lines represent nullclines of the vector field.

We generate a subdivision of the state space of the system, using the sum of squares method shown in the previous section. This gives two subdivisioning functions

$$\begin{aligned}\varphi_1(x) = &-0.000337x_1^3 + 0.00178x_1^2 - 0.00425x_1x_2 - 0.00670x_1 \\ &+ 0.000381x_2^4 + 0.00327x_2^3 + 0.0107x_2^2 + 0.0172x_2 + 0.143\end{aligned} \tag{7.82a}$$

$$\begin{aligned}\varphi_2(x) = &-0.00617x_1^3 - 0.00151x_1^2x_2 - 0.00676x_1^2 - 0.0140x_1x_2 \\ &+ 0.0150x_1 + 0.000757x_2^3 + 0.0109x_2^2 + 0.0351x_2 - 1.921.\end{aligned} \tag{7.82b}$$

Using the two subdivisioning functions, a timed automaton can be generated from the subdivisioning of the state space. We do not show the timed automaton, but the subdivision of the state space is illustrated in Figure 7.9.

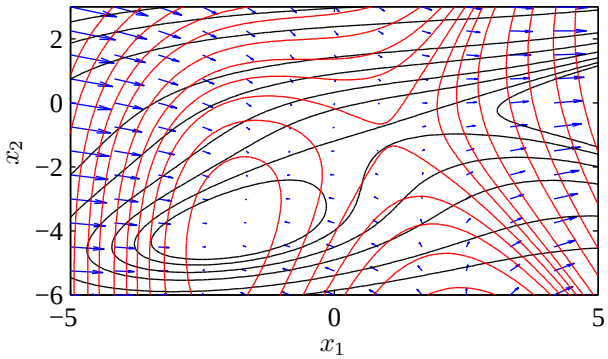


Figure 7.9: Subdivision of the state space using φ_1 (black) and φ_2 (red).

An abstraction of the system will have one location per cell in the subdivision of the state space. To allow the illustration of the timed automaton, we only show the part

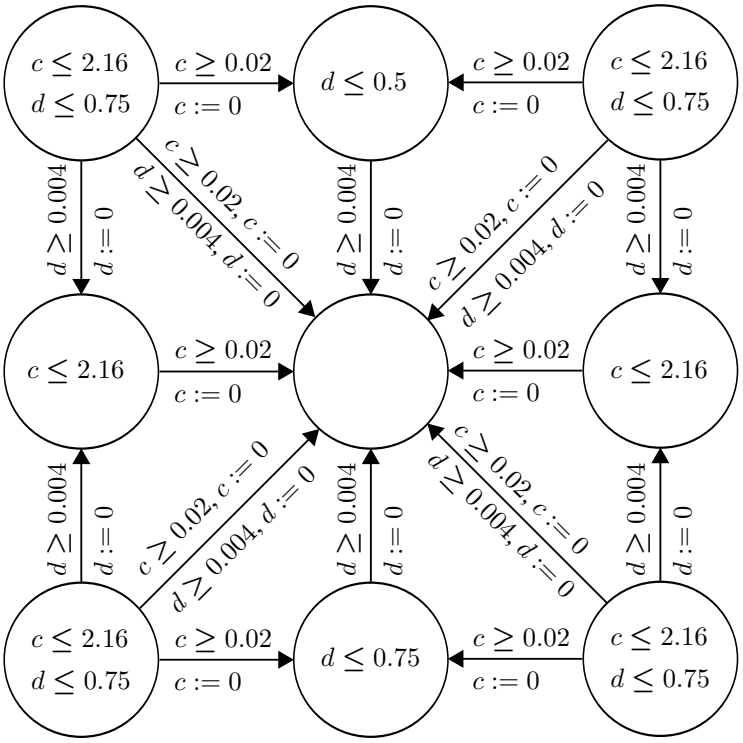


Figure 7.10: Illustration of a timed automaton abstracting the system given in (7.81).

of the abstraction that surrounds the stable critical point in $(x_1, x_2) = (-2, -4)$. The guards and invariants are generated as shown in [11], by searching for the minimum and maximum values of the derivative of the subdivisioning functions in each slice. The timed automaton is illustrated in Figure 7.10, where the middle location abstracts the cell containing the stable critical point. Therefore, no invariants are imposed in this location. The abstraction is sound - not complete. We have excluded names on the locations, to make the illustration more clear.

8 Conclusion

In this paper, a method for abstracting dynamical systems by timed automata is presented. The abstraction is based on subdividing the state space of a dynamical systems using sublevel sets of subdivisioning functions.

We show that a dynamical system can be abstracted completely by a timed automaton if the functions that generate the subdivision of the state space have directional derivatives along the vector field, that are functions of the subdivisioning function itself. This allows the verification of dynamical systems by timed automata.

Additionally, we provide algorithms for finding subdivisioning functions; however, we restrict the directional derivative of the subdivisioning function along the vector field to be an affine function of the subdivisioning function itself, and not just a polynomial

function of the subdivisioning function. The algorithms are presented as LMI and sum of squares optimization problems, which can be used to verify polynomial dynamical systems.

Finally, the method has been successfully applied on a polynomial system with two equilibria.

A Proofs and Definitions

Proof of Proposition 34. We first show that the two conditions imply that $\mathcal{A}(\mathcal{S},\mathcal{T})$ is a complete and refinable abstraction of Γ. Let $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$ then by (7.24)

$$\phi_\Gamma(t,x_0) \in S^i_{g_i} \qquad \forall t \in (0, t^i_{g_i}), \tag{7.83a}$$
$$\phi_\Gamma(t,x_0) \in S^i_{g_i} \cap S^i_{g_i-1} \qquad \forall t = t^i_{g_i}, \text{ and} \tag{7.83b}$$
$$\phi_\Gamma(t,x_0) \in S^i_{g_i-1} \qquad \forall t \in (t^i_{g_i}, t^i_{g_i} + t^i_{g_i-1}). \tag{7.83c}$$

Note that the valuation of the clock $v(c^i) = 0$ for $t = 0$; hence, by condition 2

$$\phi_\mathcal{A}(t,\alpha_K(x_0)) = \alpha_K(S^i_{g_i}) \qquad \forall t \in (0, t^i_{g_i}), \tag{7.84a}$$
$$\phi_\mathcal{A}(t,\alpha_K(x_0)) = \alpha_K(S^i_{g_i}) \cup \alpha_K(S^i_{g_i-1}) \qquad \forall t = t^i_{g_i}, \text{ and} \tag{7.84b}$$
$$\phi_\mathcal{A}(t,\alpha_K(x_0)) = \alpha_K(S^i_{g_i-1}) \qquad \forall t \in (t^i_{g_i}, t^i_{g_i} + t^i_{g_i-1}). \tag{7.84c}$$

Hence, $\alpha_K \circ \phi_\Gamma(t,x_0) = \phi_\mathcal{A}(t,\alpha_K(x_0))$.

The other direction if proved by contradiction. In contradiction to (7.24), suppose that $x'_0, x''_0 \in (\varphi^i)^{-1}(a^i_{g_i})$ and there exist $\tau' < \tau''$ such that

$$a^i_{g_i-1} = \varphi(\phi_\Gamma(\tau', x'_0)) \tag{7.85}$$
$$a^i_{g_i-1} = \phi_\Gamma(\tau'', x''_0). \tag{7.86}$$

Then there exists $\epsilon > 0$ such that

$$\phi_\Gamma(\tau', x'_0) \notin S^i_{g_i} \tag{7.87}$$
$$\phi_\Gamma(\tau'' - \epsilon, x''_0) \in S^i_{g_i}. \tag{7.88}$$

Hence, $\alpha_E \circ \phi_\Gamma(x'_0, \tau'' - \epsilon) \neq \alpha_E \circ \phi_\Gamma(x''_0, \tau'' - \epsilon)$, i.e., the abstraction cannot be complete. □

The following definitions can be found in [29] and [31].

Definition 100. An ideal I of $\mathbb{R}[\underline{x}]$ is a subset of $\mathbb{R}[\underline{x}]$ satisfying

1. $0 \in I$.
2. If $a, b \in I$ then $a + b \in I$.
3. If $a \in I$ and $b \in \mathbb{R}[\underline{x}]$, then $ab \in I$.

Definition 101. Let $S \subseteq \mathbb{R}[\underline{x}]$. The ideal generated by S is

$$I(S) = \{g_1 s_i + \cdots + g_n s_n |\ n \in \mathbb{N}, g_i \in \mathbb{R}[\underline{x}], s_i \in S\}. \tag{7.89}$$

Definition 102. For any set $S \subseteq \mathbb{R}^n$, $I(S)$ denotes the ideal in $\mathbb{R}[\underline{x}]$ consisting of all polynomials vanishing on S, i.e., $I(S) = \{f \in \mathbb{R}[\underline{x}] |\ f(x) = 0\, \forall x \in S\}$.

Definition 103. For any set S in $\mathbb{R}[\underline{x}]$, $Z(S)$ denotes the set of common zeros of S, i.e., $Z(S) = \{x \in \mathbb{R}^n | f(x) = 0\, \forall f \in S\}$.

Definition 104 (Irreducible Polynomial [28])**.** A polynomial $p \in \mathbb{R}[\underline{x}]$ is called irreducible if it is nonconstant and there exist no two nonconstant polynomials $p_1, p_2 \in \mathbb{R}[\underline{x}]$ such that

$$p = p_1 p_2. \tag{7.90}$$

Proposition 38 ([28])*: Let V be a irreducible variety and $V = \cup_{i=1}^{r} V_i$, a union of closed subsets, then $V = V_i$ for some i.*

Theorem 14 ([29])*: Let K be a real closed field, and suppose $f \in K[\underline{x}]$ is irreducible. The following are equivalent:*

1. *(f) is real.*
2. *$(f) = I(Z(f))$.*
3. *$\dim(Z(f)) = n - 1$.*
4. *The polynomial f has a non-singular zero in K^n (i.e., there is an $x \in K^n$ such that $f(x) = 0$ and $\frac{\partial f}{\partial x_i} \neq 0$ for some $i \in \boldsymbol{n}$).*
5. *The polynomial f changes sign on K^n (i.e., there exist $x, y \in K^n$ such that $f(x)f(y) < 0$).*

References

[1] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, April 1994.

[2] H. Guéguen, M.-A. Lefebvre, J. Zaytoon, and O. Nasri, "Safety verification and reachability analysis for hybrid systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 25–36, 2009.

[3] E. Asarin, T. Dang, G. Frehse, A. Girard, C. L. Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design*, Munich, Germany, 2006, pp. 1582–1587.

[4] M. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, USA, December 1998, pp. 4277–4282.

[5] O. Maler and G. Batt, "Approximating continuous systems by timed automata," in *Formal Methods in Systems Biology*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, vol. 5054, pp. 77–89.

[6] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, February 2008.

[7] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[8] C. Sloth and R. Wisniewski, "Algorithmic approach to abstracting linear systems by timed automata," in *Proceedings of the 18th IFAC World Congress*, Milano, Italy, August 2011, pp. 4546–4551.

[9] A. Abate, A. Tiwari, and S. Sastry, "Box invariance in biologically-inspired dynamical systems," *Automatica*, vol. 45, no. 7, pp. 1601–1610, 2009.

[10] R. Wisniewski and C. Sloth, "Abstraction of dynamical systems by timed automata," *Modeling, Identification and Control*, vol. 32, no. 2, pp. 79–90, 2011.

[11] C. Sloth and R. Wisniewski, "Verification of continuous dynamical systems by timed automata," *Formal Methods in System Design*, vol. 39, no. 1, pp. 47–82, 2011.

[12] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, August 2007.

[13] A. Tiwari and G. Khanna, "Nonlinear systems: Approximating reach sets," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, R. Alur and G. Pappas, Eds. Springer Berlin / Heidelberg, 2004, vol. 2993, pp. 171–174.

[14] A. Tiwari, "Abstractions for hybrid systems," *Formal Methods in System Design*, vol. 32, no. 1, pp. 57—83, 2008.

[15] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, vol. 2289, pp. 425–438.

[16] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[17] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, J. Hespanha and A. Tiwari, Eds. Springer Berlin / Heidelberg, 2006, vol. 3927, pp. 348–362.

[18] C. Belta and L. C. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1749–1759, November 2006.

[19] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[20] G. E. Bredon, *Topology and Geometry*. Springer, 1993.

[21] L. W. Tu, *An Introduction to Manifolds*. Springer, 2008.

[22] M. W. Hirsch, *Differential Topology*. Springer, Heidelberg, 1976.

[23] S. Basu, R. Pollack, and M.-F. Roy, "Complexity of computing semi-algebraic descriptions of the connected components of a semi-algebraic set," in *Proceedings of the 1998 international symposium on Symbolic and algebraic computation*. ACM, 1998, pp. 25–29.

[24] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Thomson Course Technology, 2006.

[25] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*, 2nd ed., ser. Algorithms and Computation in Mathematics. Springer, 2006, vol. 10.

[26] K. R. Meyer, "Energy functions for Morse Smale systems," *American Journal of Mathematics*, vol. 90, no. 4, pp. 1031–1040, 1968.

[27] J. W. Milnor, *Morse Theory*, ser. Annals of Mathematics Studies 51. Princeton University Press, 1963.

[28] B. Hassett, *Algebraic Geometry*. Cambridge University Press, 2007.

[29] M. Marshall, *Positive Polynomials and Sums of Squares*. American Mathematical Society, 2008, vol. 146.

[30] A. Sard, "The measure of the critical values of differentiable maps," *Bulletin American Mathematical Society*, vol. 48, no. 12, pp. 883–890, 1942.

[31] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. 293–320, 2003.

[32] J. Bochnak, M. Coste, and M.-F. Roy, *Real Algebraic Geometry*. Springer, 1998.

[33] S. Boyd and L. E. Ghaoui, "Method of centers for minimizing generalized eigenvalues," *Linear Algebra and its Applications*, vol. 188-189, pp. 63–111, 1993.

[34] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "SOSTOOLS and its control applications," in *Positive Polynomials in Control*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 2005, vol. 312, pp. 273–292.

[35] B. Gärtner and J. Matousek, *Approximation Algorithms and Semidefinite Programming*. Springer, 2012.

Paper C

Abstractions for Mechanical Systems

Christoffer Sloth and Rafael Wisniewski

This paper was published in:
Proceedings of the 4th IFAC Workshop on Lagrangian and Hamiltonian Methods
for Nonlinear Control, August 2012

Abstract

This paper proposes a method for discretizing the state space of mechanical systems. This is a first attempt in using reduction techniques for mechanical systems in the subdivisioning of the state space. The method relies on a combination of transversal and tangential manifolds for the conservative mechanical system. The tangential manifolds are generated using constants of motion, which can be derived from Noether's theorem. The transversal manifolds are subsequently generated on a reduced space, given by the Routhian, via action-angle coordinates. The method fully applies for integrable systems.

We focus on a particular aspect of abstraction - subdividing the state space, as existing methods can be applied on the discretized state space to obtain an automata-based model. The contribution of the paper is to show that well-known reduction methods can be used to generate abstract models, which can be used for formal verification.

1 Introduction

In the design of a safety-critical system, it is vital to formally verify the system before its deployment. Formal verification can be used to prove the safety of a system, i.e., that no admissible solution trajectory reaches a forbidden subset of the state space, see [1] for a survey.

There exist lots of methods for verifying different properties of systems, and the choice of a method should be done based on the dynamics of the considered system and the properties that should be verified. A method for verifying timed and temporal specifications of timed automata is presented in [2]. In addition, a framework for verifying the safety of more general stochastic hybrid systems, by the use of Lyapunov-like functions called barrier certificates, is presented in [3].

We generate an abstract model of a mechanical system based on a subdivision of the state space, to verify the system. We follow the ideas of [4, 5] and generate the subdivision using invariant sets. However, in contrast to [5] where box invariants are considered, the proposed method uses more general sets, and in contrast to [4], we provide a method for generating the tangential and transversal manifolds used in the subdivisioning for mechanical systems. To allow the verification of timed and temporal specifications, we abstract the system by a timed automaton instead of a directed graph, which is most commonly used.

In this paper, we utilize symmetry reduction techniques used in mechanics to realize the subdivision. We apply Lagrange-D'Alembert's principle to model the mechanical system. At first, we remove all dissipation and discretize the resulting conservative system. Using the Lagrangian, we identify cyclic coordinates, and generate tangential manifolds given by constants of motion of the system found via Noether's theorem. Subsequently, we generate transversal manifolds on a reduced system, given by the Routhian, via the use of action-angle coordinates. Afterwards, we add the dissipation and obtain a so-called transversal subdivision. This gives a finite discretization of the state space for *integrable mechanical systems*. For more general mechanical systems, the effectiveness of the method depends on the symmetries of the system. Note that our approach is elementary as it is accomplished in coordinates, in contrast to abstract coordinate free formalism for reductions in mechanical systems [6]. The generated abstract model is a

timed automaton that can be checked in existing tools; hence, allowing the verification of timed and temporal properties of the mechanical system. To delimit the content of the paper, we only present the method for generating the subdivision, as the abstract model subsequently can be generated using [7].

This paper is organized as follows. Section 2 contains preliminary definitions, Section 3 explains how to make abstractions for a mechanical system, and Section 4 applies the proposed subdivision on a model of the inverted pendulum on a cart. Finally, Section 5 comprises conclusions.

2 Preliminaries

The purpose of this section is to provide definitions related to dynamical systems and subdivisions of state spaces.

Dynamical System

An autonomous dynamical system $\Gamma = (X, f)$, with state space $X \subseteq \mathbb{R}^n$ and continuous map $f : X \to \mathbb{R}^n$, has dynamics described by ordinary differential equations

$$\dot{x} = f(x). \tag{8.1}$$

Let $\phi_\Gamma : [0, \epsilon] \times X_0 \to X$, $\epsilon > 0$ be the flow map satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{8.2}$$

for all $t \in [0, \epsilon]$ and $x_0 = \phi_\Gamma(0, x_0)$.

For a map $f : A \to B$, and a subset $C \subseteq A$, $f(C) \equiv \{f(x) |\ x \in C\}$. Thus, the reachable set of a system Γ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1, t_2]$ is

$$\Phi_\Gamma([t_1, t_2], X_0). \tag{8.3}$$

Subdivisioning

A definition of a cell, generated by a collection of functions is given below, where $\boldsymbol{k}$ denotes the set $\{1, \ldots, k\}$.

Definition 105 (Cell)**.** Let $\Phi = \{\varphi_i : \mathbb{R}^n \to \mathbb{R} | i \in \boldsymbol{k}\}$ be a collection of functions, let $\mathbb{A} = \{A_i | i \in \boldsymbol{k}\}$ be a collection of sets of regular values, where $A_i = \{a_i^j \in \mathbb{R} | j \in I_i \subseteq \mathbb{N}\}$ is a set of regular values of φ_i and $a_i^j < a_i^{j'}$ if $j < j'$. Assume that the level sets $\varphi_i^{-1}(a_i^j)$ and $\varphi_{i'}^{-1}(a_{i'}^{j'})$ intersect transversally for all $i \neq i'$, $j \in I_i$, and $j' \in I_{i'}$. Then a connected component of

$$\bigcap_{i=1}^{k} \varphi_i^{-1}([a_i^j, a_i^{j'}]) \tag{8.4}$$

with $a_i^j, a_i^{j'} \in A_i$ and $j < j'$ is called a cell.

A finite subdivision $E(\Phi, \mathbb{A})$ is defined to be the collection of all cells generated by Φ and $\mathbb{A}$ according to Definition 105.

Definition 106 (Transversal Subdivision)**.** Let X be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous and let $\mathrm{Cr}(f)$ be the set of critical points of f (equilibria). Let $\Phi = \{\varphi_i : X \to \mathbb{R} | i \in \boldsymbol{k}\}$ be a set of real differentiable functions, where

$$L_f\varphi(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f^j(x) \tag{8.5a}$$

and let $\mathbb{A} = \{A_i | i \in \boldsymbol{k}\}$ be a collection of sets of regular values. Then the finite subdivision $E(\Phi, \mathbb{A})$ is said to be transversal (we call it a transversal subdivision) if for each cell $e \in E(\Phi, \mathbb{A})$ there is a subdivisioning function $\varphi_i \in \Phi$ such that

$$L_f\varphi_i(x) \neq 0 \qquad \forall x \in e \backslash \mathrm{Cr}(f) \tag{8.5b}$$

and for all $i \in \boldsymbol{k}$

$$L_f\varphi_i(x) = 0 \qquad \forall x \in \mathrm{Cr}(f). \tag{8.5c}$$

It is seen from (8.5b) that at least one subdivisioning function has to have nonzero gradient in each cell (8.5b); hence, the vector field should be transversal to the level sets of at least one subdivisioning function. This is important in the generation of time information for the abstraction.

Abstraction

To motivate the proposed abstraction procedure, it is briefly explained in the following. For details see [7].

Consider a state space subdivided into a number of cells, as shown in Figure 8.1. The abstraction procedure consists of first generating discrete locations E, representing the cells. Second, an edge is added between two locations if there exists a trajectory initialized in one cell that eventually reaches the adjacent cell. Finally, time information is added as guards and invariants to quantify "eventually reached". A timed automaton is illustrated in Figure 8.1. The locations are denoted by $e_1, \ldots, e_4$; there are two clocks denoted by c and d. The transition between location e_1 and e_2 may happen whenever the clock $c \geq 2$ and must take place before the clock c exceeds 3. Once this transition occurs, the clock c resets to 0. The timed automaton $\mathcal{A}$ models the system by bounding the time that a trajectory can stay within a cell. We denote the reachable locations from initial locations E_0 on the time interval $[t_1, t_2]$ by

$$\Phi_{\mathcal{A}}([t_1, t_2], E_0). \tag{8.6}$$

To the subdivision E, we associate an abstraction function, which to each point in the state space associates the cells that this point belongs to.

Definition 107 (Abstraction Function)**.** Let $E \equiv \{e_\lambda | \ \lambda \in \Lambda\}$ be a finite subdivision of the state space X. An abstraction function for E is the multivalued function defined by

$$\alpha_E : X \to 2^E, \ \alpha_E(x) = \{e \in E | \ x \in e\}.$$

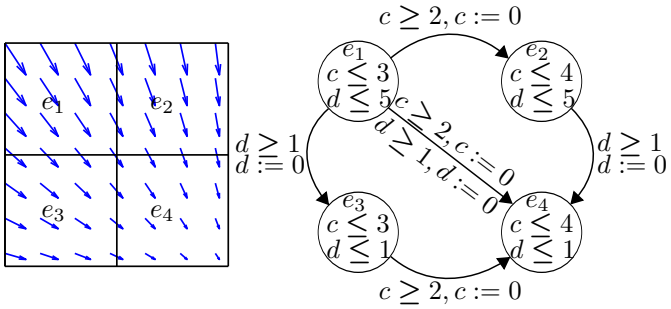


Figure 8.1: A phase plot of a system and a subdivision of its state space. The behavior of the system is abstracted by a timed automaton.

To be able to draw conclusions about the original system Γ based on the abstraction $\mathcal{A}$, it is essential to determine how the two models are related. This relation is given in terms of reachable sets in the following.

Definition 108. Let $\Gamma = (X, f)$ be a dynamical system, and suppose its state space X is subdivided by $E = \{e_1, \ldots, e_k\}$. Let the initial states $X_0 = \bigcup_{i \in \mathcal{I}} e_i$, with $\mathcal{I} \subseteq \boldsymbol{k}$. Then an abstraction $\mathcal{A}$ with locations E and initial locations $E_0 = \{e_i | i \in \mathcal{I}\}$ is said to be

1. **sound** on an interval $[t_1, t_2]$ if

$$\alpha_E \circ \Phi_\Gamma(t, X_0) \subseteq \Phi_\mathcal{A}(t, \alpha_E(X_0)), \text{ for all } t \in [t_1, t_2]$$

2. **complete** on an interval $[t_1, t_2]$ if

$$\alpha_E \circ \Phi_\Gamma(t, X_0) = \Phi_\mathcal{A}(t, \alpha_E(X_0)) \text{ for all } t \in [t_1, t_2].$$

If a sound abstraction $\mathcal{A}$ is safe then Γ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$. If a complete abstraction $\mathcal{A}$ is safe (unsafe) then Γ is also safe (unsafe).

3 Method

In this section, the method for discretizing mechanical systems is presented. The discretization is accomplished in the following steps

(A) Discard all dissipation of the system and subdivide the state space of the conservative system using tangential and transversal manifolds.

(B) Add dissipation and select level sets to obtain a transversal subdivision, see Definition 106.

(C) Generate a timed automaton abstracting the system, according to [8].

We show in Proposition 40 that a transversal subdivision generated by n subdivisioning functions can be realized for integrable systems using the presented procedure.

First, we consider the mechanical system without dissipation, as this enables the identification of cyclic coordinates and first integrals or constants of motion. The constants of motion are functions with level sets being tangential manifolds; hence, they are used as subdivisioning functions. Then the model is reduced using Routh reduction [9]. This reduced space is subdivided using transversal manifolds, which are generated via action-angle coordinates. Finally, we add dissipation to the system. This implies that the system trajectories no longer are confined to a certain constant of motion. Instead, the system trajectories traverse the manifolds according to dynamics described by the dissipation. This subdivision is shown to be transversal.

Discretizing Conservative Mechanical System

The aim of this subsection is to provide guidance for finding $2n$ mutually transversal subdivisioning functions for a conservative mechanical system with n degrees of freedom. It is required to find $2n$ mutually transversal subdivisioning functions, i.e., functions whose gradients are linearly independent at each point (except of critical points), to obtain arbitrary accuracy of the abstraction. The method consists of the following steps

1. Identify cyclic coordinates from the Lagrangian.
2. Find tangential manifolds via Noether's theorem.
3. Reduce the system using Routh reduction.
4. Find transversal manifolds for the reduced system using action-angle coordinates.

We assume that the mechanical system with n degrees of freedom is described by n Euler-Lagrange equations of motion in generalized coordinates.

Identification of Cyclic Coordinates

Recall that a coordinate q^i is said to be cyclic if the Lagrangian of a system does not depend on it.

From the Lagrangian of a system, it is seen that $\partial L/\partial q^i = 0$ if q^i is cyclic; hence, the generalized momentum $\partial L/\partial \dot{q}^i$ is constant. This means that cyclic coordinates identify symmetries of the system, where a symmetry is a transformation that generates a displacement under which the system is invariant, e.g., a translation along a cyclic coordinate. Therefore, each cyclic coordinate should be subdivided independently of the other coordinates, i.e., if q^i is a cyclic coordinate then

$$\varphi : (q, \dot{q}) \mapsto q^i \tag{8.7}$$

should be used as subdivisioning function. The cyclic coordinate should be discarded in the remainder of the subdivisioning procedure.

Identification of Tangential Manifolds

[4] proposes a partition based on tangential and transversal manifolds, generated by foliations. This method is based on the local existence of $m-1$ linear independent tangential manifolds on $\mathbb{R}^m$.

Definition 109. Suppose $N_1, \ldots, N_k$ are co-dimension 1 submanifolds of $\mathbb{R}^m$, and let $\nu(N_i, \mathbb{R}^m)$ be normal bundles of N_i (in $\mathbb{R}^m$) [10, p. 253]. Then $N_1, \ldots, N_k$ are said to be linear independent manifolds, if for any $x \in \bigcap_{i \in \boldsymbol{k}} N_i$ there exist $(x, v_i) \in \nu(N_i, \mathbb{R}^m)$, $i \in \boldsymbol{k}$ with $v_i \neq 0$, v_i are linearly independent.

It is seen that the normal to the linear independent manifolds are linearly independent at each point of their intersection. From Flow Box Theorem, see [11], it is seen that locally there exist $m-1$ tangential manifolds and one transversal manifold in the neighborhood of a regular point.

We are interested in constructing the tangential manifolds without the use of solutions of the differential equations. This motivates the identification of tangential manifolds, via the Euler-Lagrange equations. In contrary to the local analysis, the presented method may not identify $2n-1$ constants of motion; however, the tangential manifolds are identified globally. The number of constants of motion that one can find for a given system is not a priori known.

Following [12, p. 207], the function H is a first integral of the Hamiltonian phase flow with Hamiltonian function H. This implies that we can always find one constant of motion: the Hamiltonian. The Hamiltonian function should be used as a tangential subdivisioning function

$$\varphi(q, \dot{q}) = H(q, \dot{q}). \tag{8.8}$$

An integrable system has, per definition, n linear independent tangential manifolds. These are also called functionally independent constants of motion. The Poisson bracket is used in the definition of an integrable system. Recall that given two smooth real-valued functions A and B defined on the phase space of a Hamiltonian system, the canonical Poisson bracket of A and B is defined by

$$\{A, B\} = \sum_{i=1}^{N} \left(\frac{\partial A}{\partial q^i} \frac{\partial B}{\partial p_i} - \frac{\partial B}{\partial q^i} \frac{\partial A}{\partial p_i} \right), \tag{8.9}$$

where (q^i, p_i) are conjugate pairs of canonical coordinates [13].

Definition 110 (Integrable System)**.** A Hamiltonian systems in a $2n$-dimensional symplectic manifold is said to be integrable (in the Arnold-Liouville sense) if there exist n functionally independent constants of motion that are in involution, meaning that they pairwise satisfy

$$\{F_i, F_j\} = 0 \quad \forall i, j. \tag{8.10}$$

Constants of motion can be found, by using the symmetries of the system, given by the cyclic coordinates, according to the following theorem [12, p. 88].

Theorem 15 (Noether's Theorem)*: Let M be a smooth manifold, $L : TM \to \mathbb{R}$ a smooth function on its tangent bundle TM. If the system (M, L) admits the one-parameter group of diffeomorphisms $h_s : M \to M$, $s \in \mathbb{R}$, then the lagrangian system of equations corresponding to L has a first integral $I : TM \to \mathbb{R}$. In local coordinates q on M the integral I is written in the form*

$$I(q, \dot{q}) = \frac{\partial L}{\partial \dot{q}} \frac{dh_s(q)}{ds}\bigg|_{s=0}. \tag{8.11}$$

From the theorem, it is seen that we can find one constant of motion per cyclic coordinate, as the generalized momentum $\partial L/\partial \dot{q}^i$ is constant if q^i is cyclic; hence, the generalized momentum should be used as a tangential subdivisioning function

$$\varphi(q, \dot{q}) = \frac{\partial L}{\partial \dot{q}^i}. \tag{8.12}$$

In relation to Theorem 15, let $M = \mathbb{R}^n$ and let the first coordinate be a cyclic coordinate, then $h_s : (q^1, \ldots, q^n) \mapsto (q^1 + s, \ldots, q^n)$ is a one-parameter group. Note that symmetry under translation corresponds to momentum conservation, symmetry under rotation to angular momentum conservation, symmetry in time to energy conservation [14].

Reduction of the System

The remaining subdivision should be conducted on a reduced state space, given by the following theorem, which can be used to restrict the dynamics of a system to a lower dimensional surface using constants of motion, [15].

Theorem 16 (Routh Reduction)*: Let $L : \mathbb{R}^{2n} \to \mathbb{R}$ be the Lagrangian for a system with n degrees of freedom. Assume that q^1 is a cyclic coordinate and that locally $\partial^2 L/\partial \dot{q}^1 \partial \dot{q}^1 \neq 0$ so that $\dot{q}^1$ can be expressed as $\dot{q}^1 = \varrho(q^2, \ldots, q^n, \dot{q}^2, \ldots, \dot{q}^n)$. Consider the Routhian $R^\mu : \mathbb{R}^{2(n-1)} \to \mathbb{R}$ defined as the function $R^\mu = L - \dot{q}^1 \mu$, where all instances of $\dot{q}^1$ are substituted by the function ϱ and momentum $p_1 = \mu$. The Routhian is now interpreted as the Lagrangian for a system with $(n-1)$ degrees of freedom $(q^2, \ldots, q^n)$.*

Any solution $(q^1(t), \ldots, q^n(t))$ of the Euler-Lagrange equations of motion

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}^i} - \frac{\partial L}{\partial q^i} = 0, \qquad i = 1, \ldots, n \tag{8.13}$$

with momentum $p_1 = \mu$, projects onto a solution $(q^2(t), \ldots, q^n(t))$ of the Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial R^\mu}{\partial \dot{q}^k} - \frac{\partial R^\mu}{\partial q^k} = 0, \qquad k = 2, \ldots, n. \tag{8.14}$$

Conversely, any solution of the Euler-Lagrange equations for R^μ can be lifted to a solution of the Euler-Lagrange equations for L with momentum $p_1 = \mu$.

Using Theorem 16, we can obtain Euler-Lagrange equations of reduced dimension, which should be used in the generation of the transversal manifolds.

The idea of Routh reduction is to use the constants of motion as coordinates in the system description. This enables the system to be analyzed using fewer coordinates, as the system has no dynamics in the coordinates given by the constants of motion. The concept is shown in Figure 8.2.

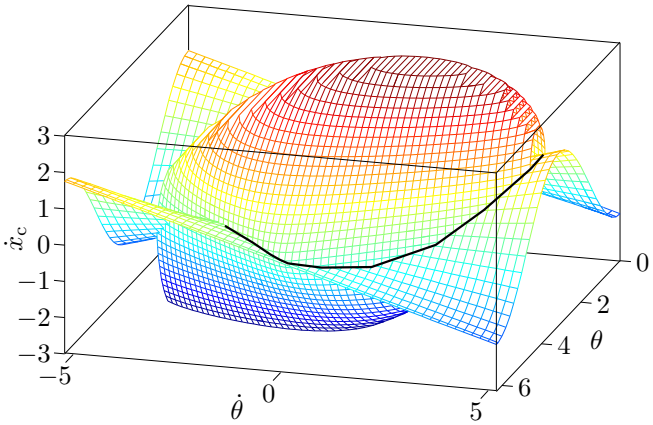


Figure 8.2: The two surfaces are level sets of the constants of motion. The black line is the simulated trajectory.

The figure illustrates two constants of motion and a solution trajectory (black line) that is located at their intersection; hence, the solution can be described using only one coordinate (apart from the constants of motion).

Identification of Transversal Manifolds

We have not found general method for finding transversal manifolds; however, for integrable systems, we can find transversal manifolds via the use of action-angle coordinates.

Theorem 17 ([16]): *Consider a completely integrable Hamiltonian system with constants of motion* $C_1(q,p) = H(q,p)$, $C_1(q,p), \ldots, C_n(q,p)$ *which are in involution. The hypersurfaces given by sets of constants* $c = \{c_i | i \in \boldsymbol{n}\}$

$$S(c) = \{(q,p) \in T^*\mathbb{Q} | C_i(q,p) = c_i, i = 1, \ldots, n\} \tag{8.15}$$

are invariant under the flow of the Hamiltonian system. If $S(c)$ *is compact and connected, then* $S(c)$ *can be mapped in a diffeomorphic way on a* n*-torus* $T^n = S^1 \times \cdots \times S^1$*. Each circle can be described by an angle coordinate* $\theta_i(t)$ *with dynamics*

$$\frac{d\theta_i}{dt} = \Omega_i(c), \qquad i = 1, \ldots, n. \tag{8.16}$$

From the theorem we see that for integrable systems one can find a coordinate system, where n coordinates are given by constants of motion and n coordinates which are independent of each other and are given by trivial dynamics. For each action-angle θ_i, a transversal subdivisioning function

$$\varphi_i(q,p) = \theta_i \tag{8.17}$$

should be used in the subdivision of the state space. For details in the synthesis of the coordinate transformation, see [16].

Note that the proposed method does not provide $2n$ linear independent subdivisioning functions for all systems; however, for integrable systems they can be found via Theorem 17. Therefore, the proposed subdivision can be applied to partly subdivide a state space, and then the remaining part of the state space can be subdivided using, e.g., hypercubes as used in most other abstraction procedures.

Obtaining Transversal Subdivision

The final step of the subdivisioning procedure is to check if the subdivision is transversal. We show that a transversal subdivision can always be found for integrable systems.

Proposition 39*: Let the system (M, L) be defined as shown in Theorem 15 with first integral in local coordinates (U, ς)*

$$I(q, \dot{q}) = \frac{\partial L}{\partial \dot{q}} \frac{dh_s(q)}{ds}. \tag{8.18}$$

Then by adding external forces Q to the system, the time derivative of I becomes

$$\frac{d}{dt} I(q, \dot{q}) = Q \frac{dh_s(q)}{ds}. \tag{8.19}$$

Proof. Let $\phi : \mathbb{R} \to U$, $q = \phi(t)$ be a local solution to the Lagrange equation. Since h_s preserves L, the translation of a solution, $h_s \circ \phi : \mathbb{R} \to U$ also satisfies the Lagrange equations for any s.

Let the mapping $\Upsilon : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^n$, given by $q = \Upsilon(s, t) = h_s(\phi(t))$. We will denote the derivatives with respect to t by dots and with respect to s by primes. By hypothesis

$$\frac{\partial L(\Upsilon, \dot{\Upsilon})}{\partial s} = \frac{\partial L}{\partial q} \Upsilon' + \frac{\partial L}{\partial \dot{q}} \dot{\Upsilon}' = 0 \tag{8.20}$$

where the partial derivatives of L are taken at the point $q = \Upsilon(s, t)$, $\dot{q} = \dot{\Upsilon}(s, t)$.

For any fixed s, $\Upsilon : \mathbb{R} \to \mathbb{R}^n$ satisfies

$$\frac{\partial L}{\partial q}(\Upsilon(s,t), \dot{\Upsilon}(s,t)) = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{q}}(\Upsilon(s,t), \dot{\Upsilon}(s,t)) \right) - Q(\Upsilon(s,t), \dot{\Upsilon}(s,t)) \tag{8.21}$$

By inserting (8.21) into (8.20) we get

$$\left(\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} \right) \Upsilon' - Q\Upsilon' + \frac{\partial L}{\partial \dot{q}} \dot{\Upsilon}' = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} q' \right) - Q\Upsilon' = 0 \tag{8.22a}$$

$$\frac{dI}{dt} = Q\Upsilon'. \tag{8.22b}$$

□

Proposition 40*: Let (M, L) be an integrable system, and let $k = 2n$; thus, $\boldsymbol{k} = \{1, \ldots, 2n\}$. Then there exists a collection of nonempty sets of regular values $\mathbb{A} = \{A_i | i \in \boldsymbol{k}\}$ for the subdivisioning functions $\Phi = \{\varphi_i(q, \dot{q}) | i \in \boldsymbol{k}\}$ (see Theorem 17), such that the generated subdivision $E = (\Phi, \mathbb{A})$ is transversal.*

Proof. Consider cells containing no critical points. Only one of the $2n$ subdivisioning functions should have a nonzero gradient in each cell. We look at points, where $dI/dt = 0$, i.e., $Q\Upsilon' = 0$. Let $\mathcal{G} = \{(q, \dot{q}) | Q\Upsilon' = 0\}$ and let $\mathrm{Cr}(f)$ be the set of critical points. By the definition of transversality, the transversal subdivisioning functions $\varphi_i(q, \dot{q})$ are transversal to the vector field at each point in $\mathcal{G} \backslash \mathrm{Cr}(f)$, i.e., for each transversal subdivisioning function given by (8.17)

$$\dot{\varphi}_i(q, \dot{q}) \neq 0 \qquad \forall (q, \dot{q}) \in \mathcal{G} \backslash \mathrm{Cr}(f). \tag{8.23}$$

This implies that there exists $\mathbb{A}$ such that the subdivision is transversal. □

4 Example

The proposed abstraction is applied to an inverted pendulum. The units are omitted to clarify the presentation.

The cart has a point mass m_c, a position x_c, velocity $\dot{x}_\mathrm{c}$, and acceleration $\ddot{x}_\mathrm{c}$. The pendulum is modeled as a point mass m_p extended from the cart in a massless rod of length l and has inertia I_p with respect to the point, where the pendulum is attached to the cart. The angle of the pendulum with respect to the vertical axis is θ. Finally, the cart is affected by a frictional force $F_\mathrm{f} = -k\dot{x}_\mathrm{c}$. A state space model of the system is shown in [17, p. 28], and we use the following parameter values: $g = 9.82$, $l = 1$, $m_\mathrm{p} = 1$, $m_\mathrm{c} = 2$, $I_\mathrm{p} = 1$, $k = 1$. The Lagrangian is

$$L(x) = \frac{1}{2}(m_\mathrm{c} + m_\mathrm{p})\dot{x}_\mathrm{c}^2 + \frac{1}{2}(I_\mathrm{p} + m_\mathrm{p} l^2)\dot{\theta}^2 + m_\mathrm{p}\dot{x}_\mathrm{c}\dot{\theta} l \cos\theta - m_\mathrm{p} g l \cos\theta. \tag{8.24}$$

Discretizing Conservative Mechanical System

Identification of Cyclic Coordinates

The cart position x_c is a cyclic coordinate, as $\partial L/\partial x_\mathrm{c} = 0$. The subdivisioning function for the cyclic coordinate is given by

$$\psi_1(x) = x_\mathrm{c}. \tag{8.25}$$

Identification of Tangential Manifolds

The Hamiltonian is a constant of motion. Therefore, we get the following subdivisioning function

$$\psi_2(x) = \frac{1}{2}(m_\mathrm{c} + m_\mathrm{p})\dot{x}_\mathrm{c}^2 + \frac{1}{2}(I_\mathrm{p} + m_\mathrm{p} l^2)\dot{\theta}^2 + m_\mathrm{p}\dot{x}_\mathrm{c}\dot{\theta} l \cos\theta + m_\mathrm{p} g l \cos\theta. \tag{8.26}$$

Second, Theorem 15 is used to identify $\partial L/\partial \dot{x}_\mathrm{c}$ as a constant of motion, since this is the conjugate momentum corresponding to x_c

$$\psi_3(x) = (m_\mathrm{c} + m_\mathrm{p})\dot{x}_\mathrm{c} + m_\mathrm{p}\dot{\theta} l \cos\theta. \tag{8.27}$$

The dimension of both $\psi_2^{-1}(a_2)$ and $\psi_3^{-1}(a_3)$ is two, and their intersection is one dimensional [18, p. 114]; hence, there exist no more linearly independent tangential manifolds. Level sets of $\psi_2(x)$ and $\psi_3(x)$ are shown in Figure 8.2.

Reduction of the System

The reduced system only depends on the variables θ and $\dot{\theta}$, and has Routhian

$$R^{\mu}(\theta,\dot{\theta}) = -\frac{1}{2}\frac{\left(\mu - m_{\mathrm{p}}\dot{\theta}l\cos\theta\right)^2}{m_{\mathrm{c}}+m_{\mathrm{p}}} + \frac{1}{2}(I_{\mathrm{p}}+m_{\mathrm{p}}l^2)\dot{\theta}^2 - m_{\mathrm{p}}gl\cos\theta \tag{8.28}$$

where $\psi_3(x) = \mu$.

Identification of Transversal Manifolds

From $R^{\mu}(\theta,\dot{\theta})$, the momentum $p_\theta \equiv \partial R^{\mu}(\theta,\dot{\theta})/\partial\dot{\theta}$ can be expressed in terms of the value of the hamiltonian function $H^{\mu}(\theta,p_\theta) = E^{\mu}$ corresponding to $R^{\mu}(\theta,\dot{\theta})$ and θ

$$p_\theta(\theta,E^{\mu}) = \pm\sqrt{2b\left(E^{\mu} + \frac{1}{2}\frac{\mu^2}{m_{\mathrm{c}}+m_{\mathrm{p}}} - m_{\mathrm{p}}gl\cos\theta\right) + a^2} \tag{8.29}$$

where

$$a = \frac{\mu m_{\mathrm{p}} l\cos\theta}{m_{\mathrm{c}}+m_{\mathrm{p}}}, \qquad b = (I_{\mathrm{p}}+m_{\mathrm{p}}l^2) - \frac{(m_{\mathrm{p}}l\cos\theta)^2}{m_{\mathrm{c}}+m_{\mathrm{p}}}.$$

The action variable is defined as

$$J = \oint p dq, \tag{8.30}$$

where the integration is over an entire period. In particular, if the pendulum takes a full rotation, then the integration is from 0 to 2π. Otherwise the integration is from the minimum angle to the maximum angle and back again. This angle can be found from (8.29), as the momentum p_θ is zero, when the pendulum reaches its minimum and maximum angle. This expression is solved in Maple, and is a third degree polynomial in $\cos(\theta)$. The expression is not explicitly shown, as it is very long, but the graph of the hamiltonian function H^{μ} as a function of θ and μ for $p_\theta = 0$ is shown in Figure 8.3.

It is seen that $\theta = \pi$ (hanging downwards), when $\mu = 0$ and $H^{\mu} = -m_{\mathrm{p}}gl$ and there are no solutions with lower H^{μ} when $\mu = 0$. Furthermore, the pendulum swings between θ and $-\theta + 2\pi$ until the value of the hamiltonian function gets greater that $m_{\mathrm{p}}gl$, where the pendulum starts to do full rotations; hence, $p_\theta \neq 0$ everywhere.

The action variable J only depends on E^{μ}, not θ. Therefore, $J = J(H^{\mu},\mu) = J(E^{\mu},\mu)$ and

$$\dot{w} = \frac{\partial H^{\mu}(J)}{\partial J} = v(J). \tag{8.31}$$

The value of v as a function of E^{μ} and μ can be calculated in Maple, and relates to frequency at which the pendulum oscillates. Finally, we get

$$w = v(J)t + \beta. \tag{8.32}$$

The action variable is not used as a subdivisioning function, as it is a constant, but $\psi_4(x) = w$ is a transversal subdivisioning function.

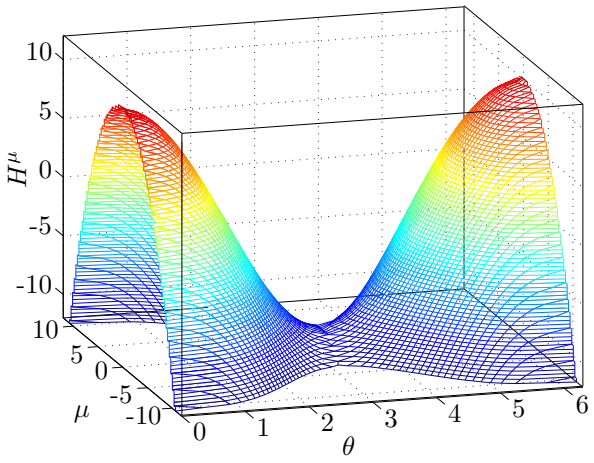


Figure 8.3: Graph of H^μ as a function of θ and μ for $p_\theta = 0$.

Generation of Abstraction

We do not show the abstraction of for the inverted pendulum, but a detailed description of the abstraction procedure, including a method for calculating the invariants and guards is shown in [8]. We use $\dot{\psi}_i(x)$

$$\dot{\psi}_1(x) = \dot{x}_c, \quad \dot{\psi}_2(x) = -k\dot{x}_c, \quad \dot{\psi}_3(x) = -k\dot{x}_c^2. \tag{8.33}$$

The value of $\dot{\psi}_4(x)$ is again calculated in Maple. Now regular values can be chosen to generate E in accordance with Proposition 40.

The generated timed automaton is a sound abstraction and can be automatically verified by a tool. Therefore, we can verify timed temporal properties of the mechanical system via the verification of the generated timed automaton in tools such as Uppaal or Kronos.

5 Conclusion

In this paper, we have provided a constructive method for subdividing the state space of integrable mechanical systems. This subdivision can be used in the abstraction of the mechanical system by a combinatorial model such as a timed automaton. The subdivision is generated by intersecting tangential and transversal manifolds. The generation of the manifolds is based on reduction techniques for mechanical systems, via the Euler-Lagrange equations and Noether's theorem. The method is applied to the inverted pendulum on a cart, showing its applicability for a nonlinear system. Furthermore, it is shown that a transversal subdivision can always be obtained for integrable systems using the proposed subdivision.

References

[1] H. Guéguen, M.-A. Lefebvre, J. Zaytoon, and O. Nasri, "Safety verification and reachability analysis for hybrid systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 25–36, 2009.

[2] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, April 1994.

[3] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, August 2007.

[4] M. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, USA, December 1998, pp. 4277–4282.

[5] A. Abate, A. Tiwari, and S. Sastry, "Box invariance in biologically-inspired dynamical systems," *Automatica*, vol. 45, no. 7, pp. 1601–1610, 2009.

[6] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry*. Springer-Verlag, 1999.

[7] C. Sloth and R. Wisniewski, "Verification of continuous dynamical systems by timed automata," *Formal Methods in System Design*, vol. 39, no. 1, pp. 47–82, 2011.

[8] R. Wisniewski and C. Sloth, "Abstraction of dynamical systems by timed automata," *Modeling, Identification and Control*, vol. 32, no. 2, pp. 79–90, 2011.

[9] H. Goldstein, *Classical Mechanics*. Addison-Wesley, 1960.

[10] J. M. Lee, *Introduction to Smooth Manifolds*. Springer, 2000.

[11] J. P. Junior and W. de Melo, *Geometric Theory of Dynamical Systems: An Introduction*. Springer, 1980.

[12] V. Arnold, *Mathematical Methods of Classical Mechanics*, 2nd ed. Springer-Verlag, 1989.

[13] J. E. Marsden, *Lectures on Mechanics*. Cambridge University Press, 1992.

[14] J. Butterfield, "On symmetry and conserved quantities in classical mechanics," *arXiv:physics/0507192v1*, 2005.

[15] B. Langerock, F. Cantrijn, and J. Vankerschaver, "Routhian reduction for quasi-invariant Lagrangians," *Journal of Mathematical Physics*, vol. 51, no. 2, 2010.

[16] J. V. Jose and E. J. Saletan, *Classical Dynamics: A Contemporary Approach*. Cambridge University Press, 1998.

[17] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[18] G. E. Bredon, *Topology and Geometry*. Springer, 1993.

Paper D

Compositional Safety Analysis using Barrier Certificates

Christoffer Sloth, George J. Pappas, and Rafael Wisniewski

This paper was published in:
Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, 2012.

The layout has been revised

Abstract

This paper proposes a compositional method for verifying the safety of a dynamical system, given as an interconnection of subsystems. The safety verification is conducted by the use of the barrier certificate method; hence, the contribution of this paper is to show how to obtain compositional conditions for safety verification.

We show how to formulate the verification problem, as a composition of coupled subproblems, each given for one subsystem. Furthermore, we show how to find the compositional barrier certificates via linear and sum of squares programming problems.

The proposed method makes it possible to verify the safety of higher dimensional systems, than the method for centrally computed barrier certificates. This is demonstrated by verifying the safety of an emergency shutdown of a wind turbine.

1 Introduction

Safety verification is an important part of developing a control system. Safety verification ensures that a control system does not violate any state constraints. Numerous methods have been developed for verifying the safety of a system; see [1] for a review. These methods range over analytical methods, numerical simulation-based methods, and discrete abstraction methods.

The safety verification determines if the reachable states of a system intersect a set of unsafe states. Computing the reachable states of a dynamical system is in general very difficult, as seen in [2]; hence, it may only be possible for systems of low dimension. Therefore, several methods have been developed to approximate the reachable set of a dynamical system. In [3], the reachable states are approximated based on simulated trajectories, by exploiting that trajectories initialized close to each other stay in the proximity of each other.

Another class of methods, e.g., [4] verifies the safety of a system, by using the vector field to find invariant sets that do not include the unsafe states. Similarly, the papers [5, 6], provide a method for calculating barrier certificates for safety analysis of continuous, stochastic, and hybrid systems. The idea of these works is to find a barrier function that is decreasing along system trajectories, and has a zero level set (a so called barrier), which no solution trajectory crosses. If the set of initial states is a subset of the zero sublevel set of the barrier function, and the set of unsafe states is in its complement, then the system is safe.

Common to the previously mentioned methods is that they verify the safety of a system, by studying a system directly. However, it may be beneficial to study a system as an interconnection of subsystems, and decompose the verification problem into smaller subproblems. This is suggested for compositional stability analysis in [7].

In this paper, we show how the barrier certificates in [5, 6] can be generated for a system, given as an interconnection of subsystems. Compositional conditions are given for finding barrier certificates. Additionally, linear matrix inequalities (LMIs) and sum of squares (SOS) are used to generate the barrier certificates, which are solved numerically, by use of SOSTOOLS for MATLAB [8].

The paper is organized as follows. Section 2 explains the verification problem in terms of barrier certificates, and Section 3 explains how to reformulate the verification problem

by a composition of certificates generated individually for each subsystem. Section 4 shows how to compute the barrier certificates, both via LMIs and polynomial inequalities. Section 5 demonstrates the use of the method, by proving safety of a shutdown procedure for a wind turbine, and Section 6 comprises conclusions.

2 Safety Verification using Barrier Certificates

In this section, we present the barrier certificate method, which can be used to verify the safety of a dynamical system.

We consider a continuous system given as a system of ordinary differential equations

$$\dot{x} = f(x, d), \tag{9.1}$$

where $x \in \mathbb{R}^n$ is the state and $d \in D \subseteq \mathbb{R}^m$ is the disturbance input.

For some measurable and essentially bounded disturbance function $\bar{d} : \mathbb{R}_{\geq 0} \to D$, i.e., $\bar{d} \in \mathcal{L}_\infty(\mathbb{R}_{\geq 0}, D)$, we denote the solution of the Cauchy problem (9.1) with $x(0) = x_0$ on an interval $[0, T]$ by $\phi^{\bar{d}}_{x_0}$, i.e.,

$$\frac{d\phi^{\bar{d}}_{x_0}(t)}{dt} = f\left(\phi^{\bar{d}}_{x_0}(t), \bar{d}(t)\right) \tag{9.2}$$

for almost all $t \in [0, T]$ and $\phi^{\bar{d}}_{x_0}(0) = x_0$.

We consider a system given by $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$, where $f : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is continuous, $X \subseteq \mathbb{R}^n$, $X_0 \subseteq X$, $X_\mathrm{u} \subseteq X$, and $D \subseteq \mathbb{R}^m$. In the safety verification, we only consider trajectories initialized in X_0 that are contained in the set X. We verify if there exists a trajectory that can reach an unsafe set X_u.

For a map $f : A \to B$ and subset $C \subset A$, we write $f(C) \equiv \{f(x) |\ x \in C\}$. Thus, the safety of a system Γ is defined as follows.

Definition 111 (Safety)**.** Let $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ be given. A trajectory $\phi^{\bar{d}}_{X_0} : [0, T] \to \mathbb{R}^n$ is unsafe if there exists a time $t \in [0, T]$ and a disturbance $\bar{d} \in \mathcal{L}_\infty(\mathbb{R}_{\geq 0}, D)$, such that $\phi^{\bar{d}}_{X_0}([0, t]) \cap X_\mathrm{u} \neq \emptyset$ and $\phi^{\bar{d}}_{X_0}([0, t]) \subseteq X$.

We say that a system Γ is safe if there are no unsafe trajectories.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{Z}(f)$ denotes the set

$$\mathcal{Z}(f) = \{x \in \mathbb{R}^n | f(x) = 0\}. \tag{9.3}$$

The safety property can be verified using the following.

Proposition 41 (Strict barrier certificate [5])*: Let $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ be given. If there exists a differentiable function $B : X \to \mathbb{R}$ satisfying*

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{9.4a}$$
$$B(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{9.4b}$$
$$L_f B(x, d) < 0 \qquad \forall (x, d) \in \mathcal{Z}(B) \times D. \tag{9.4c}$$

Then the system Γ is safe.

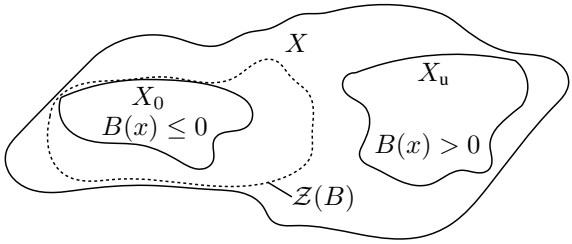


Figure 9.1: Illustration of a set X, which contains the initial set X_0 and the unsafe set X_u. The dashed line illustrates the zero level set of B.

Proposition 41 states that a trajectory initialized within the zero sublevel set of a function B, cannot cross the zero level set $\mathcal{Z}(B)$, if B is decreasing (along system trajectories) on the zero level set. This is illustrated in Figure 9.1.

The set of barrier certificates satisfying Proposition 41 is nonconvex, due to (9.4c). However, the following more conservative proposition has a convex set of feasible barrier certificates. The convexity property becomes apparent in the computation of the barrier certificates in Section 4.

Corollary 9 (Weak barrier certificate [5, 6])*: Let* $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ *be given. If there exists a differentiable function* $B : X \to \mathbb{R}$ *satisfying*

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{9.5a}$$
$$B(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{9.5b}$$
$$L_f B(x, d) \leq 0 \qquad \forall (x, d) \in X \times D. \tag{9.5c}$$

Then the system Γ *is safe.*

Corollary 9 states that a trajectory of a system initialized in a state within the zero sublevel set of a nonincreasing function (along system trajectories), cannot reach the complement of the zero sublevel set.

The difference between Proposition 41 and Corollary 9 is that (9.5c), in contrast to (9.4c), must hold for all states and all disturbances. Additionally, the inequality constraint (9.4c) is strict weathers it is weak in (9.5c).

3 Compositional Barrier Certificates

In this section, we assume that a dynamical system is given as an interconnection of subsystems. This allows the safety verification to be split up into smaller subproblems in addition to some coupling constraints.

To provide an overview of the proposed compositional setup, we initially consider an example from [7], consisting of three interconnected subsystems. The interconnection of the three subsystems is shown in Figure 9.2. Properties of the interconnected system are to be analyzed by studying its components as isolated systems, in conjunction with their coupling.

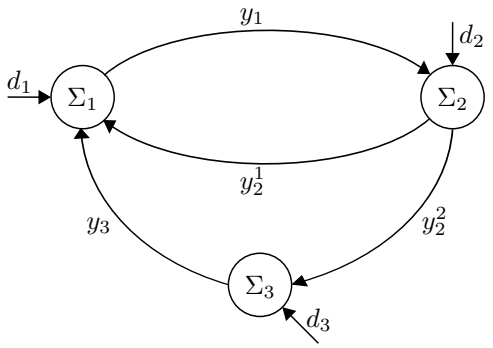


Figure 9.2: Interconnection of three subsystems $\Sigma_1, \Sigma_2, \Sigma_3$.

Let each subsystem be described by a system of continuous ordinary differential equations and an output map

$$\Sigma_1 : \begin{cases} \dot{x}_1 = f_1(x_1, d_1, u_1) \\ y_1 = h_1(x_1) \end{cases} \tag{9.6a}$$

$$\Sigma_2 : \begin{cases} \dot{x}_2 = f_2(x_2, d_2, u_2) \\ y_2 = h_2(x_2) \end{cases} \tag{9.6b}$$

$$\Sigma_3 : \begin{cases} \dot{x}_3 = f_3(x_3, d_3, u_3) \\ y_3 = h_3(x_3), \end{cases} \tag{9.6c}$$

where $x_i \in X_i \subseteq \mathbb{R}^{n_i}$ is the state, $d_i \in D_i \subseteq \mathbb{R}^{m_i}$ is the disturbance, and $u_i \in \mathbb{R}^{q_i}$ is an interconnection input, given by $u_i = g_i(x_1, \ldots, \hat{x}_i, \ldots, x_k)$. Here, $\hat{x}_i$ indicates that x_i is removed. Additionally, $y_i \in \mathbb{R}^{r_i}$ is an interconnection output, given by the map $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{r_i}$. Note that the interconnection of the subsystems gives a relation between u_i and y_i. In Figure 9.2, $y_2 = (y_2^1, y_2^2)$, $u_1 = (y_2^1, y_3)$, $u_2 = y_1$, and $u_3 = y_2^2$.

The output y_i belongs to the set

$$Y_i \equiv h_i(X_i) \subseteq \mathbb{R}^{r_i}. \tag{9.7a}$$

Similarly, u_i belongs to the set

$$U_i \equiv g_i(X_1, \ldots, \hat{X}_i, \ldots, X_k) \subseteq \mathbb{R}^{q_i}. \tag{9.7b}$$

In the remainder of the paper, we present a method for generating barrier certificates for some general topology of the interconnection of subsystems.

Let $k \in \mathbb{N}$ be the number of subsystems. For $i = 1, \ldots, k$ we consider a system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$, where $\{f_i\}$ is a collection of continuous vector fields with $f_i : \mathbb{R}^{n_i+m_i+q_i} \to \mathbb{R}^{n_i}$, $X_i \subseteq \mathbb{R}^{n_i}$, $X_{0,i}, X_{\mathrm{u},i} \subseteq X_i$, and $D_i \subseteq \mathbb{R}^{m_i}$. Let

$$X = X_1 \times \cdots \times X_k \subseteq \mathbb{R}^n, \tag{9.8a}$$

$$X_0 = X_{0,1} \times \cdots \times X_{0,k} \subseteq X, \tag{9.8b}$$

$$X_{\mathrm{u}} = X_{\mathrm{u},1} \times \cdots \times X_{\mathrm{u},k} \subseteq X, \tag{9.8c}$$

$$D = D_1 \times \cdots \times D_k \subseteq \mathbb{R}^m, \tag{9.8d}$$

$$U = U_1 \times \cdots \times U_k \subseteq \mathbb{R}^q, \text{ and} \tag{9.8e}$$
$$Y = Y_1 \times \cdots \times Y_k \subseteq \mathbb{R}^r. \tag{9.8f}$$

Remark 15*: The assumption that the sets X and D are given as cartesian products of X_i and D_i in* (9.8)*, limits the sets that can be directly expressed; however, by using multiple sets, the original set can, in principle, be approximated. Therefore, the previous restriction does not theoretically restrict the method, but it may complicate the computations involved in the safety verification.*

In the following, we present two lemmas that show how to compose the inequality constraints on the barrier function and its derivative in Proposition 41 into separate constraints for the subsystems and coupling constraints. We omit the proofs of both lemmas, as they are straightforward.

In Lemma 9, we let the vector field be given as an interconnection of subsystems, and show that (9.4c) can be composed into an inequality constraint for each subsystem, and a coupling constraint.

Lemma 9. *Let $k \in \mathbb{N}$. Let $x = (x_1, \ldots, x_k) \in X$, $d = (d_1, \ldots, d_k) \in D$, $u = (u_1, \ldots, u_k) \in U$, $y = (y_1, \ldots, y_k) \in Y$, where X, D, U, Y are given as shown in* (9.8)*. For $i = 1, \ldots, k$ let*

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} f_1(x_1, d_1, u_1) \\ \vdots \\ f_k(x_k, d_k, u_k) \end{bmatrix} = f(x, d), \tag{9.9a}$$
$$u_i = g_i(x_1, \ldots, \hat{x}_i, \ldots, x_k), \tag{9.9b}$$
$$y_i = h_i(x_i). \tag{9.9c}$$

Suppose that there is a bijective map $\Upsilon : U \to Y$.

Then there exists a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that

$$\varphi(x) f(x, d) < 0 \qquad \forall (x, d) \in X \times D \tag{9.10}$$

if for $i = 1, \ldots, k$ there exist continuous functions $\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and $\gamma_i : \mathbb{R}^{q_i + r_i} \to \mathbb{R}$ such that for all $(x_i, d_i, u_i) \in X_i \times D_i \times U_i$

$$\varphi_i(x_i) f_i(x_i, d_i, u_i) < \gamma_i(u_i, h(x_i)) \text{ and} \tag{9.11a}$$
$$\sum_i \gamma_i(u_i, h(x_i)) \leq 0. \tag{9.11b}$$

Lemma 9 can be used to decompose (9.4c) into an inequality constraint for each subsystem in addition to a coupling constraint.

Lemma 10. *Let $k \in \mathbb{N}$. For $i = 1, \ldots, k$ let $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ be a continuous function, and $X_i \subseteq \mathbb{R}^{n_i}$ be compact. There exists a constant $c_i \in \mathbb{R}$ for all i such that*

$$f_i(x_i) - c_i \leq 0 \qquad \forall x_i \in X_i \text{ and} \tag{9.12a}$$
$$\sum_i c_i \leq 0 \qquad \forall x_i \in X_i \tag{9.12b}$$

if and only if

$$\sum_i f_i(x_i) \leq 0 \qquad \forall x_i \in X_i. \tag{9.13}$$

Proof. See Paper E on page 196. □

Using Lemma 9 and Lemma 10, we rewrite Proposition 41 as follows.

Proposition 42*: Let $k \in \mathbb{N}$ and let the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$ be given. If there exist differentiable functions $B_i : X_i \to \mathbb{R}$, constants $\alpha_i, \beta_i \in \mathbb{R}$, and functions $\gamma_i : \mathbb{R}^{q_i+r_i} \to \mathbb{R}$ for $i = 1, \ldots, k$ such that*

$$B_i(x_i) + \alpha_i \leq 0 \;\; \forall x_i \in X_{0,i}, \tag{9.14a}$$
$$B_i(x_i) - \beta_i > 0 \;\; \forall x_i \in X_{\mathrm{u},i}, \tag{9.14b}$$
$$L_{f_i} B_i(x_i, d_i, u_i) < \gamma_i(u_i, h_i(x_i)) \qquad \forall u_i \in U_i,\, x_i \in \mathcal{Z}(B_i),\, d_i \in D_i, \tag{9.14c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \sum_i \gamma_i(u_i, h_i(x_i)) \leq 0 \qquad \forall u_i \in U_i, x_i \in \mathcal{Z}(B_i). \tag{9.14d}$$

Then the system Γ is safe.

Proof. We show that Proposition 42 ensures that the conditions in Proposition 41 are satisfied. Let $x \equiv (x_1, \ldots, x_k)^{\mathsf{T}}$ and $B : \mathbb{R}^{n_1+\cdots+n_k} \to \mathbb{R}$ be defined as $B(x) = \sum_i B_i(x_i)$. By Lemma 10, (9.14a) and (9.14b) are by the satisfaction of (9.14d) equivalent to

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{9.15a}$$
$$B(x) > 0 \qquad \forall x \in X_{\mathrm{u}}. \tag{9.15b}$$

Finally, by (9.14c) and (9.14d)

$$\sum_i L_{f_i} B_i(x_i, d_i, u_i) < \sum_i \gamma_i(u_i, h_i(x_i)) \leq 0 \qquad \forall u_i \in U_i,\, x_i \in \mathcal{Z}(B_i),\, d_i \in D_i. \tag{9.15c}$$

This is by Lemma 9 equivalent to (9.4c). Hereby, the system Γ is safe. □

The inequality constraints (9.14a)-(9.14c) must be satisfied for each subsystem, and (9.14d) couples the subproblems. Notice that the function B is decreasing along the solution, but each function B_i is not necessarily decreasing along the solution.

In the following, we rewrite Corollary 9 using the same technique.

Corollary 10*: Let $k \in \mathbb{N}$ and let the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$ be given. If there exist differentiable functions $B_i : X_i \to \mathbb{R}$, constants $\alpha_i, \beta_i \in \mathbb{R}$, and functions $\gamma_i : \mathbb{R}^{q_i+r_i} \to \mathbb{R}$ for $i = 1, \ldots, k$ such that*

$$B_i(x_i) + \alpha_i \leq 0 \qquad \forall x_i \in X_{0,i}, \tag{9.16a}$$
$$B_i(x_i) - \beta_i > 0 \qquad \forall x_i \in X_{\mathrm{u},i}, \tag{9.16b}$$
$$L_{f_i} B_i(x_i, d_i, u_i) \leq \gamma_i(u_i, h_i(x_i)) \qquad \forall u_i \in U_i,\, x_i \in X_i,\, d_i \in D_i, \tag{9.16c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \sum_i \gamma_i(u_i, h_i(x_i)) \leq 0 \qquad \forall u_i \in U_i, x_i \in X_i. \tag{9.16d}$$

Then the system Γ is safe.

Proposition 42 and Corollary 10 provide compositional conditions for the safety verification. In the next section, we show how to compute the barrier certificates.

4 Computation of Barrier Certificates

In this section, we show how to compute barrier certificates from the conditions set up in Section 2 and Section 3.

Remark that any desired computational method may be applied to find the barrier certificates, and that different methods can be applied on different subproblems for the compositional conditions in Section 3. This is beneficial if some subsystems are linear and others are polynomial.

To demonstrate the computation of barrier certificates, we show how to compute the barrier certificates using sum of squares programming and linear programming. The primary focus is on sum of squares programming, as it is a generalization of linear programming. Therefore, we only explicitly formulate LMI conditions for the solution of Corollary 10.

To do the computations in a tool such as MATLAB, we restrict the vector fields to be linear (for linear programs) and polynomial (for sum of squares programs). Furthermore, we parameterize the barrier certificates as polynomials, respectively quadratic forms, and require the invariant, initial, unsafe, and disturbance sets to be given by linear and polynomial equality or inequality constraints.

First, we set up some notation about polynomials.

Definition 112 (Polynomial [9])**.** A polynomial p in n variables $x_1, \ldots, x_n$ is a finite linear combination of monomials

$$p(x) = \sum_\alpha c_\alpha x^\alpha = \sum_\alpha c_\alpha x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}, \tag{9.17}$$

where $c_\alpha \in \mathbb{R}$ and the sum is over a finite number of n-tuples $\alpha = [\alpha_1, \ldots, \alpha_n]$ with $\alpha_i \geq 0$.

The total degree of a monomial x^α is $\alpha_1 + \cdots + \alpha_n$. Additionally, the total degree of a polynomial is equal to the highest degree of its component monomials. The degree of a polynomial p is denoted by $\deg(p)$.

We only consider polynomials with real valued variables, and denote the set of polynomials in n variables by $\mathcal{P}_n$. Recall that a map $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be polynomial if its coordinate functions are polynomials, i.e., $f_i \in \mathcal{P}_n$ for $i = 1, \ldots, m$; hence, $f \in \mathcal{P}_n^m$.

Sum of squares polynomials are used in the generation of safety certificates and are explained in the following, based on [9].

Definition 113. A polynomial $p \in \mathcal{P}_n$ is called sum of squares (SOS) if

$$p = \sum_{i=1}^{k} p_i^2 \tag{9.18}$$

for some polynomials $p_i \in \mathcal{P}_n$ with $i = 1, \ldots, k$.

We denote the set of sum of squares polynomials in n variables by Σ_n.

The set of sum of squares polynomials is a subset of nonnegative polynomials [9], which can be treated using semidefinite programming, as described below.

The existence of a sum of squares decomposition of a polynomial $p \in \mathcal{P}_n$, with $d = \deg(p)$, can be expressed as a semidefinite programming feasibility problem. Therefore, the formulation of a problem as sum of squares makes the problem computationally tractable; however, the number of decision variables in the program is

$$N = \binom{n+2d}{2d} = \frac{(n+2d)!}{2d!n!}. \tag{9.19}$$

In the search for sum of squares polynomials, it is exploited that the existence of a SOS decomposition of a polynomial p is equivalent to the existence of a positive semidefinite matrix $Q = Q^{\mathrm{T}} \geq 0$ such that

$$p = Z^{\mathrm{T}} Q Z, \tag{9.20}$$

where Z is a vector of monomials of degree less than or equal to half the degree of p.

Let $k, l \in \mathbb{N}$, let $\alpha_{i,j} \in \mathcal{P}_n$ for $(i,j) \in \{1, \ldots, l\} \times \{1, \ldots, k\}$, and $w_j \in \mathbb{R}$. An SOS programming problem is

$$\underset{(c_1,\ldots,c_k)\in\mathbb{R}^k}{\text{minimize}} \sum_{j=1}^{k} w_j c_j \text{ subject to} \tag{9.21a}$$

$$\alpha_{i,0} + \sum_{j=1}^{k} \alpha_{i,j} c_j \in \Sigma_n \forall i = 1, \ldots, l. \tag{9.21b}$$

It is seen that an SOS programming problem is a minimization of a linear cost, subject to SOS feasibility constraints.

Computation of Barrier Certificates

To compute barrier certificates using sum of squares programming, we restrict the vector fields to be polynomial. Furthermore, the invariant, initial, unsafe, and disturbance sets must be semialgebraic sets, i.e., be given by polynomial inequalities, as follows.

Let $g_X : \mathbb{R}^n \to \mathbb{R}^{k_X}$, $g_{X_0} : \mathbb{R}^n \to \mathbb{R}^{k_{X_0}}$, $g_{X_\mathrm{u}} : \mathbb{R}^n \to \mathbb{R}^{k_{X_\mathrm{u}}}$, and $g_D : \mathbb{R}^m \to \mathbb{R}^{k_D}$ for some $k_X, k_{X_0}, k_{X_\mathrm{u}}, k_D \in \mathbb{N}$ be given as vectors of polynomials $g_i \in \mathcal{P}_n$, i.e., for example $g_X \in \mathcal{P}_n^{k_X}$ and $g_X = [g_1, \ldots, g_{k_X}]^{\mathrm{T}}$. Then

$$X \equiv \{x \in \mathbb{R}^n | g_X(x) \geq 0\}, \tag{9.22a}$$

$$X_0 \equiv \{x \in \mathbb{R}^n | g_{X_0}(x) \geq 0\}, \tag{9.22b}$$

$$X_\mathrm{u} \equiv \{x \in \mathbb{R}^n | g_{X_\mathrm{u}}(x) \geq 0\}, \tag{9.22c}$$

$$D \equiv \{d \in \mathbb{R}^m | g_D(d) \geq 0\}, \tag{9.22d}$$

where the inequalities in (9.22) are satisfied *entry-wise*.

Example 16. We show how (9.22) can be used to form a cylindrical set. Let $x \in \mathbb{R}^3$, $x_{1,\min}, x_{1,\max}, x_{2,c}, x_{3,c}, r \in \mathbb{R}$ and g_X be

$$g_X(x) = \begin{bmatrix} (x_1 - x_{1,\min})(x_{1,\max} - x_1) \\ r^2 - (x_2 - x_{2,c})^2 - (x_3 - x_{3,c})^2 \end{bmatrix}. \tag{9.23}$$

It is seen that $g_X(x) \geq 0$, when $x_1 \in [x_{1,\min}, x_{1,\max}]$ and (x_2, x_3) is in the disk centered at $(x_{2,c}, x_{3,c})$ with radius r. This implies that the set X given by (9.22a) and (9.23) is a cylinder.

In the computation of barrier certificates, we use a generalization of the $\mathcal{S}$-procedure [10], which is shown in Lemma 11.

Lemma 11. *Let V be a subset of $X \subseteq \mathbb{R}^n$. Let $f \in \mathcal{P}_n$ and $g \in \mathcal{P}_n^k$. Suppose $g(x) \geq 0$ (element-wise) for any $x \in V$. If*

1. *$\lambda \in \Sigma_n^k$ and*
2. *$f - \lambda^{\mathrm{T}} g \in \Sigma_n$.*

Then $f(x) \geq 0$ for all $x \in V$.

Now we can compute barrier certificates that satisfy Proposition 41 using sum of squares.

Proposition 43: Let the system $\Gamma = (f, X, X_0, X_u, D)$ and polynomials g_* shown in (9.22) be given, and let $\epsilon_1, \epsilon_2 > 0$. If there exist $B \in \mathcal{P}_n$, $\lambda_{X_0} \in \Sigma_n^{k_{X_0}}$, $\lambda_{X_u} \in \Sigma_n^{k_{X_u}}$, $\lambda_B \in \mathcal{P}_{n+m}$, and $\lambda_D \in \Sigma_{n+m}^{k_D}$ such that

$$-B - \lambda_{X_0}^{\mathrm{T}} g_{X_0}, \tag{9.24a}$$

$$B - \epsilon_1 - \lambda_{X_u}^{\mathrm{T}} g_{X_u}, \text{ and} \tag{9.24b}$$

$$-L_f B - \epsilon_2 - \lambda_D^{\mathrm{T}} g_D - \lambda_B^{\mathrm{T}} B \tag{9.24c}$$

are sum of squares. Then the system Γ is safe.

As Proposition 43 follows directly from Proposition 20 in [5], no proof is provided. However, all conditions follow directly from Lemma 11. Consider (9.24a), where $B \in \mathcal{P}_n$, $g_{X_0} \in \mathcal{P}_n^{k_{X_0}}$, $\lambda_{X_0} \in \Sigma_n^{k_{X_0}}$, (9.24a)$\in \Sigma_n$, and $g_{X_0}(x) \geq 0$ for any $x \in X_0$. Then $B(x) \leq 0$ for all $x \in X_0$.

Note that (9.24c) contains a scalar product between λ_B and B, which are both unknown. This is the reason why the conditions in Proposition 43 cannot be found directly by an SOS programming problem, neither by a linear program for quadratic B. Therefore, we generate an iterative algorithm for solving the problem. This algorithm is similar to iterative algorithms used for solving bilinear matrix inequalities via LMIs, see [11].

In the following iterative algorithm, it is necessary to get a feasible solution in each step. Therefore, the barrier certificate is initially found for only a subset of the disturbances $\tilde{D} \subseteq D$, initial conditions $\tilde{X}_0 \subseteq X_0$, etc., to ease the feasibility. Let $c_X \in \mathbb{R}_{\geq 0}^{k_X}$

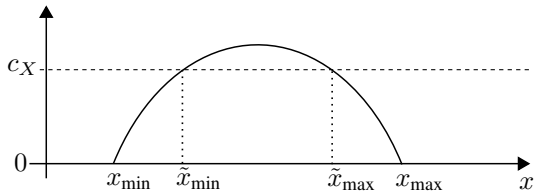


Figure 9.3: Illustration of g_X and the sets $X = [x_{\min}, x_{\max}]$ and $\tilde{X} = [\tilde{x}_{\min}, \tilde{x}_{\max}]$, given by g_X and $\tilde{g}_X$.

be a vector of nonnegative numbers. Let $\tilde{g}_X = g_X - c_X$ and define

$$\tilde{X} \equiv \{x \in \mathbb{R}^n | \tilde{g}_X(x) \geq 0\} \subseteq X. \tag{9.25}$$

By decreasing each entry of c_X, the set $\tilde{X}$ is enlarged, and if $c_X = 0$ then $\tilde{X} = X$. This is illustrated in Figure 9.3 for a set given by $g_X(x) = (x - x_{\min})(x_{\max} - x)$.

It is seen that the map g_X generates the set $X = [x_{\min}, x_{\max}]$ and $\tilde{X} = [\tilde{x}_{\min}, \tilde{x}_{\max}]$. If c_X is greater than the maximum value of g_X, then $\tilde{X} = \emptyset$.

Algorithm 1*: Let the system* $\Gamma = (f, X, X_0, X_{\mathrm{u}}, D)$ *and polynomials* g_* *shown in* (9.22) *be given.*

0. ***Initialization****: Choose vectors* $c_{X_0} \in \mathbb{R}^{k_{X_0}}_{\geq 0}$, $c_{X_{\mathrm{u}}} \in \mathbb{R}^{k_{X_{\mathrm{u}}}}_{\geq 0}$, $c_D \in \mathbb{R}^{k_D}_{\geq 0}$ *such that each entry* $c_{i,*}$ *is sufficiently large and define polynomials* $\tilde{g}_* \equiv g_* - c_*$*. Choose* $\epsilon_1, \epsilon_2 > 0$ *and specify a polynomial* $\lambda_B \in \mathcal{P}_{n+m}$*, e.g., by choosing* $\lambda_B = 0$ *or* 1*. Find* $B \in \mathcal{P}_n$, $\lambda_{X_0} \in \Sigma^{k_{X_0}}_n$, $\lambda_{X_{\mathrm{u}}} \in \Sigma^{k_{X_{\mathrm{u}}}}_n$*, and* $\lambda_D \in \Sigma^{k_D}_{n+m}$ *such that*

$$-B - \lambda^{\mathrm{T}}_{X_0} \tilde{g}_{X_0}, \tag{9.26a}$$

$$B - \epsilon_1 - \lambda^{\mathrm{T}}_{X_{\mathrm{u}}} \tilde{g}_{X_{\mathrm{u}}}, \text{ and} \tag{9.26b}$$

$$-L_f B - \epsilon_2 - \lambda^{\mathrm{T}}_D \tilde{g}_D - \lambda^{\mathrm{T}}_B B \tag{9.26c}$$

are sum of squares.

1. ***Fix the barrier certificate****: Fix* B *obtained from the previous step. Choose vectors* $\Delta c_* \geq 0$ *and update* c_**, such that* $c_* := c_* - \Delta c_*$ *and redefine the polynomials* $\tilde{g}_* \equiv g_* - c_*$*. Find* $\lambda_{X_0} \in \Sigma^{k_{X_0}}_n$, $\lambda_{X_{\mathrm{u}}} \in \Sigma^{k_{X_{\mathrm{u}}}}_n$, $\lambda_B \in \mathcal{P}_{n+m}$*, and* $\lambda_D \in \Sigma^{k_D}_{n+m}$ *such that* (9.26) *are sum of squares.*

2. ***Fix multiplier****: Fix* λ_B *obtained in the previous step. Choose vectors* $\Delta c_* \geq 0$ *and update* c_**, such that* $c_* := c_* - \Delta c_*$ *and redefine the polynomials* $\tilde{g}_* \equiv g_* - c_*$*. Find* $B \in \mathcal{P}_n$, $\lambda_{X_0} \in \Sigma^{k_{X_0}}_n$, $\lambda_{X_{\mathrm{u}}} \in \Sigma^{k_{X_{\mathrm{u}}}}_n$*, and* $\lambda_D \in \Sigma^{k_D}_{n+m}$ *such that* (9.26) *are sum of squares.*
   If all entries of the vector c_* *are zero, then terminate the algorithm; otherwise, go to step 1.*

If Algorithm 1 terminates, then Γ is safe.

Algorithm 1 alternates between freezing the coefficients of λ_B and B, to remove the product between the two unknown polynomials in (9.26c). Furthermore, the set of disturbances and the sets for which B should be positive or negative are initially smaller than D, X_0 and X_{u} and are gradually enlarged until they are equal to X_0 and X_{u}. In the enlargement of the sets, it is important that a feasible solution is found in each step of the algorithm. Notice that Algorithm 1 is not guaranteed to terminate or converge to

the global optimum; however, this is a general problem with non-convex optimization problems, see e.g. [12].

Corollary 9 can be solved directly, via the following SOS programming problem.

Corollary 11*: Let the system* $\Gamma = (f, X, X_0, X_\mathrm{u}, D)$ *and polynomials* g_* *shown in* (9.22) *be given, and let* $\epsilon_1 > 0$*. If there exist* $B \in \mathcal{P}_n$*,* $\lambda_{X_0} \in \Sigma_n^{k_{X_0}}$*,* $\lambda_{X_\mathrm{u}} \in \Sigma_n^{k_{X_\mathrm{u}}}$*,* $\lambda_X \in \Sigma_{n+m}^{k_X}$*, and* $\lambda_D \in \Sigma_{n+m}^{k_D}$ *such that*

$$-B - \lambda_{X_0}^\mathrm{T} g_{X_0}, \tag{9.27a}$$

$$B - \epsilon_1 - \lambda_{X_\mathrm{u}}^\mathrm{T} g_{X_\mathrm{u}}, \text{ and} \tag{9.27b}$$

$$-L_f B - \lambda_X^\mathrm{T} g_X - \lambda_D^\mathrm{T} g_D \tag{9.27c}$$

are sum of squares. Then the system Γ *is safe.*

Computation of Compositional Barrier Certificates

In this subsection, we show how barrier certificates can be expressed in a compositional manner, using SOS optimization for Proposition 42 and Corollary 10, and using LMIs for Corollary 10. The interconnected system can be formulated as one system, but this would increase the number of decision variables involved in the safety verification, compared to the proposed compositional approach. This is an important issue when working with SOS optimization, and is apparent from (9.19).

Let $k \in \mathbb{N}$ be the number of subsystems, and define $g_* \in \mathcal{P}^{k_*}$. In the decomposition, the considered sets are restricted, as shown in (9.8), where

$$X_i \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_i}(x_i) \geq 0\}, \tag{9.28a}$$

$$X_{0,i} \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_{0,i}}(x_i) \geq 0\}, \tag{9.28b}$$

$$X_{\mathrm{u},i} \equiv \{x_i \in \mathbb{R}^{n_i} | g_{X_{\mathrm{u},i}}(x_i) \geq 0\}, \tag{9.28c}$$

$$D_i \equiv \{d_i \in \mathbb{R}^{m_i} | g_{D_i}(d_i) \geq 0\}, \tag{9.28d}$$

$$U_i \equiv \{u_i \in \mathbb{R}^{q_i} | g_{U_i}(u_i) \geq 0\}. \tag{9.28e}$$

Proposition 42 is written in terms of SOS in the following.

Proposition 44*: Let* $k \in \mathbb{N}$*, the polynomials* g_* *shown in* (9.28)*, and the dynamical system* $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$ *be given, and let* $\epsilon_1, \epsilon_2 > 0$*. If there exist* $B_i \in \mathcal{P}_{n_i}$*,* $\alpha_i \in \mathbb{R}$*,* $\beta_i \in \mathbb{R}$*,* $\gamma_i \in \mathcal{P}_{q_i+r_i}$*,* $\lambda_{X_{0,i}} \in \Sigma_{n_i}^{k_{X_{0,i}}}$*,* $\lambda_{X_{\mathrm{u},i}} \in \Sigma_{n_i}^{k_{X_{\mathrm{u},i}}}$*,* $\lambda_{B_i} \in \mathcal{P}_{n_i+m_i+q_i}$*,* $\lambda_{D_i} \in \Sigma_{n_i+m_i+q_i}^{k_{D_i}}$*, and* $\lambda_{U_i} \in \Sigma_{n_i+m_i+q_i}^{k_{U_i}}$ *such that*

$$-B_i - \lambda_{X_{0,i}}^\mathrm{T} g_{X_{0,i}} - \alpha_i, \tag{9.29a}$$

$$B_i - \epsilon_1 - \lambda_{X_{\mathrm{u},i}}^\mathrm{T} g_{X_{\mathrm{u},i}} - \beta_i, \text{ and} \tag{9.29b}$$

$$-L_{f_i} B_i - \epsilon_2 + \gamma_i - \lambda_{D_i}^\mathrm{T} g_{D_i} - \lambda_{B_i}^\mathrm{T} B_i - \lambda_{U_i}^\mathrm{T} g_{U_i} \tag{9.29c}$$

are sum of squares and

$$\sum_i \alpha_i, \sum_i \beta_i, \text{ and } -\sum_i \gamma_i \tag{9.29d}$$

are sum of squares. Then the system Γ *is safe.*

Proposition 44 has a product between $\lambda_{B_i}^{\mathrm{T}}$ and B_i, which implies that Algorithm 1 must be used to solve it. Additionally, dual decomposition should be used to decompose the conditions; however, this is only demonstrated for the following SOS program for solving Corollary 10.

Corollary 12*: Let $k \in \mathbb{N}$, the polynomials g_* shown in* (9.28)*, and the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$ be given, and let $\epsilon_1 > 0$. If there exist $B \in \mathcal{P}_{n_i}$, $\alpha_i \in \mathbb{R}$, $\beta_i \in \mathbb{R}$, $\gamma_i \in \mathcal{P}_{q_i+r_i}$, $\lambda_{X_{0,i}} \in \Sigma_{n_i}^{k_{X_{0,i}}}$, $\lambda_{X_{\mathrm{u},i}} \in \Sigma_{n_i}^{k_{X_{\mathrm{u},i}}}$, $\lambda_{X_i} \in \Sigma_{n_i+m_i+q_i}^{k_{X_i}}$, $\lambda_{D_i} \in \Sigma_{n_i+m_i+q_i}^{k_D}$, and $\lambda_{U_i} \in \Sigma_{n_i+m_i+q_i}^{k_{U_i}}$ such that*

$$-B_i - \lambda_{X_{0,i}}^{\mathrm{T}} g_{X_{0,i}} - \alpha_i, \tag{9.30a}$$

$$B_i - \epsilon_1 - \lambda_{X_{\mathrm{u},i}}^{\mathrm{T}} g_{X_{\mathrm{u},i}} - \beta_i, \text{ and} \tag{9.30b}$$

$$-L_{f_i} B_i + \gamma_i - \lambda_{X_i}^{\mathrm{T}} g_{X_i} - \lambda_{D_i}^{\mathrm{T}} g_{D_i} - \lambda_{U_i}^{\mathrm{T}} g_{U_i} \tag{9.30c}$$

are sum of squares and

$$\sum_i \alpha_i, \sum_i \beta_i, \text{ and } -\sum_i \gamma_i. \tag{9.30d}$$

are sum of squares. Then the system Γ is safe.

In the following, we show how to prove safety using LMIs based on Corollary 10. The vector field f_i is given by

$$\dot{x} = A_i x_i + B_{1,i} d_i + B_{2,i} u_i \tag{9.31a}$$

$$y_i = C_i x_i, \tag{9.31b}$$

where A_i is an $n_i \times n_i$ matrix, $B_{1,i}$ is an $n_i \times m_i$ matrix, $B_{2,i}$ is an $n_i \times m_i$ matrix, and C_i is an $q_i \times n_i$ matrix. We say that $B_i(x_i) = x_i^{\mathrm{T}} P_i x_i$, $g_*(x_i) = x_i^{\mathrm{T}} G_* x_i$ where P_i and G_* are symmetric matrices, and $\alpha_i, \beta_i \in \mathbb{R}$. Furthermore, we define γ_i as

$$\gamma_i = \begin{bmatrix} u_i \\ x_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \Gamma_{u,i} & 0 \\ 0 & \Gamma_{x,i} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \end{bmatrix}, \tag{9.32}$$

where $\Gamma_{u,i}$ and $\Gamma_{x,i}$ are diagonal matrices.

Corollary 13*: Let $k \in \mathbb{N}$, the polynomials g_* shown in* (9.28)*, and the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\}, \{D_i\})$ be given, where $\{f_i\}$ is a collection of linear vector fields, and G_* is symmetric. If there exist $P_i = P_i^{\mathrm{T}}$, $\alpha_i, \beta_i \in \mathbb{R}$, and matrices $\Gamma_{u,i}, \Gamma_{x,i}$ given in* (9.32)*, $\lambda_{X_{0,i}} \in \mathbb{R}_{\geq 0}$, $\lambda_{X_i} \in \mathbb{R}_{\geq 0}$, $\lambda_{D_i} \in \mathbb{R}_{\geq 0}$, and $\lambda_{U_i} \in \mathbb{R}_{\geq 0}$ such that*

$$-P_i - \lambda_{X_{0,i}} G_{X_{0,i}} - \alpha_i I \geq 0 \tag{9.33a}$$

$$P_i - \lambda_{X_i} G_{X_i} - \beta_i I > 0, \text{ and} \tag{9.33b}$$

$$\begin{bmatrix} A_i^{\mathrm{T}} P_i + P_i A_i + \lambda_{X_i} G_{X_i} & P_i B_{1,i} & P_i B_{2,i} & C^{\mathrm{T}} \\ B_{1,i}^{\mathrm{T}} P_i & \lambda_{D_i} G_{D_i} & 0 & 0 \\ B_{2,i}^{\mathrm{T}} P_i & 0 & \lambda_{U_i} G_{U_i} - \Gamma_{u,i} & 0 \\ C & 0 & 0 & -\Gamma_{x,i} \end{bmatrix} < 0, \tag{9.33c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \sum_i \gamma_i \leq 0. \tag{9.33d}$$

Then the system Γ is safe.

To practically solve the safety problem in Corollary 12, we set up an optimization problem by use of dual decomposition [13]. Dual decomposition can be used to solve different types of optimization problems. We consider only the following type of optimization problem.

$$\begin{aligned} &\text{minimize } f(x) = f_1(x_1, y) + f_2(x_2, y) \text{ subject to} \\ &x_1 \in C_1,\, x_2 \in C_2,\, h_1(x_1, y) + h_2(x_2, y) \leq 0. \end{aligned} \tag{9.34}$$

We decompose the optimization problem (9.34) into two separate optimization problems, which are coupled through some additional decision variables as follows

$$\begin{aligned} &\text{minimize } f(x) = f_1(x_1, y_1) + f_2(x_2, y_2) \text{ subject to} \\ &x_1 \in C_1,\, x_2 \in C_2,\, y_1 = y_2,\, h_1(x_1, y_1) + h_2(x_2, y_2) \leq 0. \end{aligned}$$

The dual problem can be set up, as f_1 and f_2 have no shared variables. The Lagrangian for the problem is

$$\begin{aligned} L(x_1, y_1, x_2, y_2, \lambda_1, \lambda_2) = f_1(x_1, y_1) + f_2(x_2, y_2) \\ +\lambda_1^{\mathrm{T}}(y_1 - y_2) + \lambda_2 \left(h_1(x_1, y_1) + h_2(x_2, y_2)\right). \end{aligned} \tag{9.35}$$

Let $\lambda = (\lambda_1, \lambda_2)$. The dual function becomes

$$\varphi(\lambda_1, \lambda_2) = \varphi_1(\lambda_1, \lambda_2) + \varphi_2(\lambda_1, \lambda_2), \tag{9.36}$$

where

$$\varphi_1(\lambda) = \inf_{x_1, y_1} \left(f_1(x_1, y_1) + \lambda_1^{\mathrm{T}} y_1 + \lambda_2 h_1(x_1, y_1)\right), \tag{9.37a}$$

$$\varphi_2(\lambda) = \inf_{x_2, y_2} \left(f_2(x_2, y_2) - \lambda_1^{\mathrm{T}} y_2 + \lambda_2 h_2(x_2, y_2)\right). \tag{9.37b}$$

The optimization problems for φ_1 and φ_2 can be solved independently, given values for λ_1 and λ_2. Finally, the master problem is

$$\text{maximize } \varphi_1(\lambda_1, \lambda_2) + \varphi_2(\lambda_1, \lambda_2), \tag{9.38}$$

with variables λ_1 and λ_2.

To solve the master problem, we utilize the subgradient algorithm given in [14]. Note that all functions in this paper are polynomial, thus differentiable; hence, other gradient methods can be used instead of the subgradient method.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, and let $x, y \in \mathbb{R}^n$. Then any vector $g \in \mathbb{R}^n$ that satisfies

$$f(y) \geq f(x) + g^{\mathrm{T}}(y - x) \tag{9.39}$$

is called a subgradient at x.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then the subgradient algorithm gives a sequence of points $\{x^{(k)}\}_{k=0}^{\infty}$ according to

$$x^{(k+1)} = x^{(k)} - \Delta_k g^{(k)}, \tag{9.40}$$

where $x^{(k)}$ is the k^{th} iterate, $x^{(0)}$ is the initial point, $g^{(k)}$ is a subgradient of f at $x^{(k)}$, and Δ_k is the step size. When the function f to be minimized is differentiable, then $g^{(k)}$ is the unique gradient of f at point $x^{(k)}$.

For diminishing step size, the algorithm is guaranteed to converge to the optimal value, see [15]. Therefore, we use the following diminishing step size

$$\Delta_k = \frac{a}{b+k}, \tag{9.41}$$

where $a > 0$ and $b \geq 0$.

The following algorithm is used to solve the dual decomposition for the problem shown in (9.34). Note that we denote by $\bar{x}_1^{(k)}$ and $\bar{y}_1^{(k)}$ the optimal values of x_1 and y_1 for problem (9.37a) at iteration k, given some λ_1, λ_2.

Algorithm 2*: Given an optimization problem, as shown in* (9.34)*.*

0. ***Initialization****: Let $k = 0$, define the step size Δ_k, and choose some $\lambda_1^{(0)}$, $\lambda_2^{(0)}$, $\epsilon > 0$.*
1. ***Solve subproblems****:*
   Solve (9.37a) *to find $\bar{x}_1^{(k)}$ and $\bar{y}_1^{(k)}$,*
   solve (9.37b) *to find $\bar{x}_2^{(k)}$ and $\bar{y}_2^{(k)}$.*
2. ***Update dual variables****:*
   $\lambda_1^{(k+1)} := \lambda_1^{(k)} - \Delta_k(\bar{y}_2^{(k)} - \bar{y}_1^{(k)})$,
   $\lambda_2^{(k+1)} := \lambda_2^{(k)} + \Delta_k(h_1(\bar{x}_1^{(k)}, \bar{y}_1^{(k)}) + h_2(\bar{x}_2^{(k)}, \bar{y}_2^{(k)}))$,
   $k := k + 1$.
   If $|\lambda_1^{(k+1)} - \lambda_1^{(k)}| > \epsilon$, then go to step 1. Otherwise, terminate the algorithm.

Note that step 2 in Algorithm 2 tries to maximize (9.38).

The first observation in the considered problem is that γ_i has to be a diagonal matrix; otherwise, the cost of the optimization problem is not linear. For convenience, we let $\bar{\gamma}_i$ be a vector containing the diagonal elements of γ_i. Let $\lambda \equiv (\lambda_1, \lambda_2, \lambda_3)$ the dual function is

$$\varphi(\lambda) = \sum_i \varphi_i(\lambda), \tag{9.42}$$

where

$$\varphi_i(\lambda) \equiv \inf_{\alpha_i, \beta_i, \bar{\gamma}_i} -\lambda_1 \alpha_i - \lambda_2 \beta_i + \lambda_3^{\mathrm{T}} \bar{\gamma}_i \tag{9.43}$$

subject to

$$-B_i - \lambda_{X_{0,i}}^{\mathrm{T}} g_{X_{0,i}} - \alpha_i, \tag{9.44a}$$

$$B_i - \epsilon_1 - \lambda_{X_{\mathrm{u},i}}^{\mathrm{T}} g_{X_{\mathrm{u},i}} - \beta_i, \text{ and} \tag{9.44b}$$

$$-L_{f_i} B_i + \gamma_i - \lambda_{D_i}^{\mathrm{T}} g_{D_i} \tag{9.44c}$$

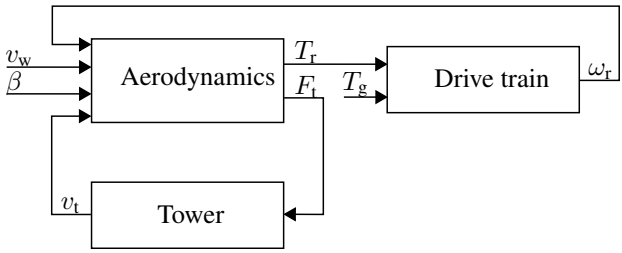


Figure 9.4: Wind turbine modeled as an interconnection of three subsystems.

are sum of squares.

Remark that λ_3 is a vector. The dual problem becomes

$$\sup_{\lambda \geq 0} \sum_i \varphi_i(\lambda). \tag{9.45}$$

In the following, we explain how the subgradient algorithm can be used to solve the previous optimization problem. Let $\alpha_i^*(\lambda)$ be the optimal value of α_i for a given λ. Then the gradients of $\varphi_1(\lambda), \ldots, \varphi_k(\lambda)$ are

$$g_i(\lambda) = \begin{bmatrix} \alpha_i^*(\lambda) & \beta_i^*(\lambda) & \gamma_i^*(\lambda) \end{bmatrix}. \tag{9.46}$$

From (9.39) and (9.46), we get for all $\mu \equiv (\mu_1, \mu_2, \mu_3)$ and $i = 1, \ldots, k$

$$\varphi_i(\mu) \geq \varphi_i(\lambda) + g_i(\mu - \lambda). \tag{9.47}$$

The function to be maximized is $\varphi(\lambda) = \sum_i \varphi_i(\lambda)$, which has a gradient $g(\lambda^{(k)}) = \sum_i g_i(\lambda^{(k)})$. The vector of multipliers is updated according to (9.40), and is

$$\lambda^{(k+1)} = \lambda^{(k)} - \Delta_k g^{\mathrm{T}} \left(\lambda^{(k)} \right). \tag{9.48}$$

It is seen that if $\sum_i \alpha_i \geq 0$ is violated, then $\lambda_1^{(k+1)} > \lambda_1^{(k)}$, as the first element of $g(\lambda^{(k)})$ is negative. This puts a larger penalty on the violation of the constraint through the dual variable λ_1.

5 Example

In this section, we demonstrate the applicability of the compositional safety analysis, by analyzing the safety of an emergency shutdown of a wind turbine. The emergency shutdown procedure is simplified for presentation, and the wind turbine model is a slight modification of the CART3 wind turbine model [16].

The wind turbine is modeled as an interconnection of three subsystems: aerodynamics (subsystem 1), tower (subsystem 2), and drive train (subsystem 3). A block diagram of the wind turbine is shown in Figure 9.4.

The wind turbine is driven by an exogenous input - the wind v_w. Via the aerodynamics, the wind exerts a torque T_r on the rotor shaft, and a force F_t on the top of the tower.

This bends the tower and makes the rotor shaft rotate. The rotor shaft is connected to a generator through a gear and a generator shaft. A converter applies a torque T_g to the generator shaft.

The magnitude of the torque T_r and the force F_t depends on the pitch angle β, the rotor speed ω_r, and the wind speed at the rotor $v_\mathrm{w} - v_\mathrm{t}$, given by the speed of the wind v_w and the velocity of the tower v_t. These relations are usually described by lookup tables (C_p and C_t tables); however, we approximate them by polynomials.

In case of severe faults, a wind turbine is shut down by pitching the blades to an angle of $\beta = 90^\circ$, while applying a constant generator torque $T_\mathrm{g} = 3,580$ Nm, until the rotor speed is below a threshold of 0.77 rad/s, from which it is not possible to apply a torque from the generator; hence, the wind turbine is left uncontrolled. At a pitch angle of 90°, the aerodynamic thrust is acting in the opposite direction of the nominal rotation; hence, it decelerates. Additionally, by applying a relatively high generator torque, the rotor shaft is decelerated even faster. This may cause the tower to sway too much or twist the rotor shaft beyond the limit accepted by the turbine structure. Therefore, we verify that this does not happen. The subsystems of the wind turbine are modeled as shown in (9.49), and will be left without further explanation.

$$\begin{aligned}
\begin{bmatrix} \dot{v}_\mathrm{r} \\ \dot{\omega}_\mathrm{r,f} \end{bmatrix} &= \begin{bmatrix} -c_{v_r} v_\mathrm{r} + (v_\mathrm{w} - v_\mathrm{t}) \\ -c_{\omega_\mathrm{r,f}} \omega_\mathrm{r,f} + \omega_\mathrm{r} \end{bmatrix} \\
h_1 &= \begin{bmatrix} p_1 \\ p_2 \end{bmatrix},
\end{aligned} \tag{9.49a}$$

$$\begin{aligned}
\begin{bmatrix} \dot{v}_\mathrm{t} \\ \dot{x}_\mathrm{t} \end{bmatrix} &= \begin{bmatrix} \frac{1}{M_\mathrm{t}}(F_\mathrm{t} - B_\mathrm{t} v_\mathrm{t} - k_\mathrm{t} x_\mathrm{t}) \\ v_\mathrm{t} \end{bmatrix} \\
h_2 &= v_\mathrm{t}
\end{aligned} \tag{9.49b}$$

$$\begin{aligned}
\begin{bmatrix} \dot{\omega}_\mathrm{r} \\ \dot{\theta}_\Delta \\ \dot{\omega}_\mathrm{g} \end{bmatrix} &= \begin{bmatrix} \frac{1}{J_\mathrm{r}}(T_\mathrm{r} - k_\mathrm{r}\theta_\Delta - B_\mathrm{r}(\omega_\mathrm{r} - \frac{1}{N_\mathrm{g}}\omega_\mathrm{g})) \\ \omega_\mathrm{r} - \frac{1}{N_\mathrm{g}}\omega_\mathrm{g} \\ \frac{1}{J_\mathrm{g}}\left(\frac{1}{N_\mathrm{g}}(k_\mathrm{r}\theta_\Delta + B_\mathrm{r}(\omega_\mathrm{r} - \frac{1}{N_\mathrm{g}}\omega_\mathrm{g})) - T_\mathrm{g}\right) \end{bmatrix} \\
h_3 &= \omega_\mathrm{r}
\end{aligned} \tag{9.49c}$$

where

$$p_1 = \left(c_{11} + c_{12}\omega_\mathrm{r,f} + c_{13} v_\mathrm{r} + c_{14}\omega_\mathrm{r,f}^2 + c_{15} v_\mathrm{r}^2 + c_{16}\omega_\mathrm{r,f} v_\mathrm{r}\right) v_\mathrm{r}^3 \tag{9.49d}$$

$$p_2 = (c_{21} + c_{22}\omega_\mathrm{r,f} + c_{23} v_\mathrm{r} + c_{24} v_\mathrm{r}^2 + c_{25}\omega_\mathrm{r,f} v_\mathrm{r}) v_\mathrm{r}^2 + c_{26} + c_{27}\omega_\mathrm{r,f}^2 \tag{9.49e}$$

The parameters of the wind turbine are the following: $M_\mathrm{t} = 7.76 \cdot 10^3$ kg, $B_\mathrm{t} = 18.6$ kN/(m/s), $k_\mathrm{t} = 2.7$ MN/m, $N_\mathrm{g} = 43$, $J_\mathrm{r} = 611.1 \cdot 10^3$ kgm^2, $B_\mathrm{r} = 24$ kNm/(rad/s), $k_\mathrm{r} = 24.7 \cdot 10^6$ Nm/rad, $c_{v_r} = 11.65$, $c_{\omega_\mathrm{r,f}} = 21$, $c_{11} = -32.42 \cdot 10^6$, $c_{12} = -746.0 \cdot 10^6$, $c_{13} = 53.03{\cdot}10^6$, $c_{14} = -1.128{\cdot}10^9$, $c_{15} = -18.63{\cdot}10^6$, $c_{16} = 384.6{\cdot}10^6$, $c_{21} = 8492.6$, $c_{22} = 300.88 \cdot 10^3$, $c_{23} = -11.85 \cdot 10^3$, $c_{24} = 3584.0$, $c_{25} = -90.32 \cdot 10^3$, $c_{26} = 318.3$, and $c_{27} = 1.692 \cdot 10^6$. We have omitted the units on the constants c_*, as they have no physical interpretation.

The considered region of the state space is

$$X_1 = [2,\ 28] \times [0.77,\ 4], \tag{9.50a}$$
$$X_2 = [-0.01,\ 0.07] \times [-0.05,\ 0.05], \tag{9.50b}$$
$$X_3 = [0.77,\ 4] \times [-25,\ 25] \cdot 10^{-3} \times [33.2,\ 172.7]. \tag{9.50c}$$

Furthermore, the inputs to the subsystems take values in the following sets

$$D_1 = [5,\ 25], \tag{9.51a}$$
$$U_1 = [0.77,\ 4] \times [-0.5,\ 0.5], \tag{9.51b}$$
$$U_2 = [1.3554,\ 94.413] \cdot 10^3, \tag{9.51c}$$
$$U_3 = [-141.86,\ -0.126] \cdot 10^6. \tag{9.51d}$$

It is chosen to initialize the system in the so called full load region, corresponding to a wind speed between 11.7 m/s and 25 m/s, where the wind turbine is operated at a constant rotor speed of 3.88 rad/s; hence, the set of initial states is

$$X_{0,1} = [11.7,\ 25] \times [3.8,\ 3.95], \tag{9.52a}$$
$$X_{0,2} = [-0.005,\ 0.005] \times [0.01,\ 0.02], \tag{9.52b}$$
$$X_{0,3} = [3.8,\ 3.95] \times [6.25,\ 6.27] \cdot 10^{-3} \times [164,\ 171]. \tag{9.52c}$$

We should verify that the following unsafe sets cannot be reached

$$X_{\mathrm{u},1} = [2,\ 3] \cup [27,\ 28] \times [0.77,\ 4], \tag{9.53a}$$
$$X_{\mathrm{u},2} = [-0.01,\ 0] \cup [0.06,\ 0.07] \times [-0.04,\ -0.03] \cup [0.03,\ 0.04], \tag{9.53b}$$
$$X_{\mathrm{u},3} = [0.77,\ 4] \times [-25,\ -10] \cdot 10^{-3} \cup [10,\ 25] \cdot 10^{-3} \times [33.2,\ 172.7]. \tag{9.53c}$$

Now the verification problem has been set up, and we do the verification using Corollary 12. To allow the verification, we need to characterize X_i, $X_{0,i}$, $X_{\mathrm{u},i}$, and U_i by polynomials. This is accomplished as by specifying a maximum value $x_{\max}$ and minimum value $x_{\min}$ of some variable x, and then defining

$$g \equiv -(x - x_{\min})(x - x_{\max}). \tag{9.54}$$

The polynomial g is nonnegative for $x \in [x_{\min},\ x_{\max}]$ and otherwise negative.

To give an impression of the convergence of the algorithm, the values of the multipliers $\lambda_1, \ldots, \lambda_6$ are shown in Figure 9.5. The safety of the system is verified by Corollary 12, and the barrier function is

$$\begin{aligned} B(x) = {} & 0.0388\omega_{\mathrm{r}}^2\theta_\Delta^2 + 0.0350\omega_{\mathrm{r}}^2\theta_\Delta + 0.748\omega_{\mathrm{r}}^2 - 0.00869\omega_{\mathrm{r}}\omega_{\mathrm{g}} + 0.569\omega_{\mathrm{g}}^2 \\ & - 0.00332\omega_{\mathrm{g}}\theta_\Delta + 1.223\theta_\Delta^2 - 97.0 \cdot 10^{-6} v_{\mathrm{t}}^2 x_{\mathrm{t}} - 0.256 v_{\mathrm{t}}^2 \\ & + 0.00173 v_{\mathrm{t}} x_{\mathrm{t}}^2 - 2.15 v_{\mathrm{t}} x_{\mathrm{t}} + 0.0658 v_{\mathrm{t}} - 0.755 x_{\mathrm{t}}^2 + 0.0785 x_{\mathrm{t}} \\ & + 0.00387 v_{\mathrm{r}}^2 - 0.00943 v_{\mathrm{r}}\omega_{\mathrm{r,f}} - 0.107 v_{\mathrm{r}} + 0.0207\omega_{\mathrm{r,f}}^2 + 0.103\omega_{\mathrm{r,f}} \\ & + 1.609. \end{aligned} \tag{9.55}$$

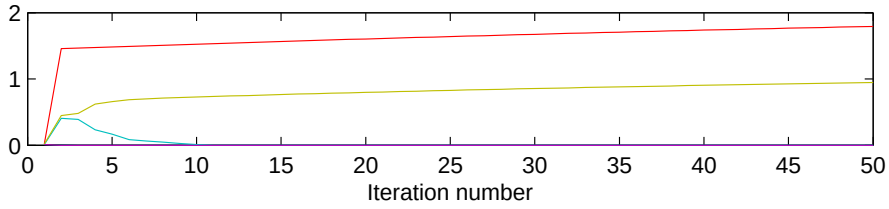


Figure 9.5: Values of the multipliers λ as a function of the number of iterations.

6 Conclusion

We have presented a method for verifying the safety of an interconnection of subsystems. The method is based on the identification of barrier certificates, where the certificates are found for each subsystem, but are coupled through some additional constraints.

The presented method allows the safety verification of higher dimensional systems, as the verification is decomposed into smaller coupled subproblems, and allows subsystems to be analyzed with different computational methods.

The method has been used to verify the safety of an emergency shutdown procedure for a wind turbine.

References

[1] H. Guéguen, M.-A. Lefebvre, J. Zaytoon, and O. Nasri, "Safety verification and reachability analysis for hybrid systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 25–36, 2009.

[2] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[3] A. Girard and G. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3927, pp. 272–286.

[4] A. Abate, A. Tiwari, and S. Sastry, "Box invariance in biologically-inspired dynamical systems," *Automatica*, vol. 45, no. 7, pp. 1601–1610, 2009.

[5] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, August 2007.

[6] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 2993, pp. 271–274.

[7] U. Topcu, A. Packard, and R. Murray, "Compositional stability analysis based on dual decomposition," in *Proceedings of the 48th IEEE Conference on Decision and Control*, December 2009, pp. 1175–1180.

[8] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "SOSTOOLS and its control applications," in *Positive Polynomials in Control*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 2005, vol. 312, pp. 273–292.

[9] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. 293–320, 2003.

[10] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. SIAM studies in applied mathematics. SIAM, 1994, vol. 15.

[11] J. Helton and O. Merino, "Coordinate optimization for bi-convex matrix inequalities," in *Proceedings of the 36th IEEE Conference on Decision and Control*, December 1997, pp. 3609–3613.

[12] T. Iwasaki, "The dual iteration for fixed-order control," *IEEE Transactions on Automatic Control*, vol. 44, no. 4, pp. 783–788, April 1999.

[13] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods," 2008.

[14] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski, *Minimization methods for non-differentiable functions*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[15] B. T. Polyak, "Subgradient methods: A survey of Soviet research," in *Proceedings of a IIASA Workshop*, ser. Nonsmooth Optimization, vol. 3, 1977, pp. 5–29.

[16] J. Laks, L. Pao, and A. Wright, "Control of wind turbines: Past, present, and future," in *Proceedings of the 2009 American Control Conference*, June 2009, pp. 2096–2103.

Paper E

On the Existence of Compositional Barrier Certificates

Christoffer Sloth, Rafael Wisniewski, and George J. Pappas

This paper was published in:
Proceedings of the 51st IEEE Conference on Decision and Control, December 2012.

The paper is modified to accommodate the assessment committee's corrections.

Abstract

This paper provides a necessary and sufficient condition for the compositional verification of a continuous system with additively separable barrier functions. The compositional safety verification enables the verification of an interconnection of subsystems. The idea behind the compositional analysis is to allow the verification of systems with a high dimension, by the verification of multiple lower dimensional subproblems. In the compositional safety analysis, a particular structure is imposed on the barrier certificate, restricting the applicability of the method.

Some examples that can be verified to be safe using the centralized method cannot be verified using the compositional method. These examples highlight how not to decompose systems, and should be used to guide the decomposition of a system into appropriate subsystems. Finally, we provide a refined condition for the compositional safety analysis that enables the verification of the more general systems, by imposing a less restrictive structure of the barrier function, but the refined method has an increased computational complexity.

1 Introduction

Safety verification is a necessary part of developing safety-critical control systems, where a malfunction may have severe consequences. The safety verification ensures that a control system does not violate any state constraints. Numerous methods have been developed for verifying the safety of a system; see [1] for a survey. These methods range over analytical methods, numerical simulation-based methods, and discrete abstraction methods.

The safety verification determines if the reachable set intersects a set of unsafe states. The computation of the reachable states for a dynamical system is in general very difficult [2], and it may only be possible to calculate approximate the reachable states for systems of low dimension. According to [3], safety verification is applicable for systems with approximately five or less continuous states. Therefore, several methods have been developed to approximate the reachable set of a dynamical system. In [4], the reachable states are approximated using a finite number of simulated trajectories, and exploiting an incremental stability condition.

Another class of methods, e.g., [5, 6] verifies the safety of a system, by using the vector field to find invariant sets that do not include the unsafe states. Similarly, the papers [7, 8] provide a method for calculating barrier certificates for safety analysis of continuous, stochastic, and hybrid systems. The idea of these works is to find a barrier function that is decreasing along system trajectories, and has a zero level set (a so-called barrier), which no solution trajectory crosses. If the set of initial states is a subset of the zero sublevel set of the barrier function, and the set of unsafe states is in its complement, then the system is safe.

The generation of the barrier certificates is similar to the generation of Lyapunov functions for proving stability. Therefore, it is important to use a computational method that scales well. Therefore, linear matrix inequalities (LMIs) and sum of squares (SOS) are used to generate the barrier certificates [9].

Common to the previously mentioned methods is that they verify the safety of a system, by studying a system directly. However, it may be beneficial to study a system as an interconnection of subsystems, and decompose the verification problem into smaller subproblems. This is suggested for compositional stability analysis in [10] and an analysis framework based on assume-guarantee reasoning is presented in [11]. In addition, a compositional method for generating barrier certificates is proposed in [12].

In this paper, we show when compositional barrier certificates can be generated using the method presented in [12]. It is shown that barrier certificates generated by the compositional method are additively separable functions. This implies that the decomposition of a system into subsystems should be generated such that an additively separable barrier certificate exists. Otherwise, the compositional method fails to verify the system. This is a very restrictive assumption; hence, the method is not as general as the centralized method presented in [7, 8]. However, in oppose the centralized method, the compositional method scales very well.

To alleviate potential issues with the compositional method, we propose another safety condition that is capable of handling more systems, but the method has a higher computational complexity.

To shorten the presentation, we only consider so-called weak barrier certificates, but the results apply for strict barrier certificates as well. Furthermore, we do not show how to algorithmically generate the certificates. Details about the generation of barrier certificates can be found in [12].

The paper is organized as follows. Section 2 explains the verification problem in terms of barrier certificates, and Section 3 explains the compositional condition for generating barrier certificates. Section 4 classifies the barrier certificates that can be generated by the compositional method, and Section 5 proposes a more general method for doing the compositional safety analysis. Finally, Section 6 comprises conclusions.

2 Barrier Certificates

In this section, we present the barrier certificate method, which can be used to verify the safety of a dynamical system.

We consider a continuous system given as a system of ordinary differential equations

$$\dot{x} = f(x), \tag{10.1}$$

where $x \in \mathbb{R}^n$ is the state. Compared to [7, 8, 12] on which this paper is based, the disturbance input to the system is omitted to clarify the presentation. However, the results in this paper can be easily extended to include disturbances.

We denote the solution of the Cauchy problem (10.1) with $x(0) = x_0$ on an interval $[0, T]$ by ϕ_{x_0}, i.e.,

$$\frac{d\phi_{x_0}(t)}{dt} = f\left(\phi_{x_0}(t)\right) \tag{10.2}$$

for all $t \in [0, T]$.

We consider a system given by $\Gamma = (f, X, X_0, X_\mathrm{u})$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuous, $X \subseteq \mathbb{R}^n$, $X_0 \subseteq X$, and $X_\mathrm{u} \subseteq X$. In the safety verification, we only consider

trajectories initialized in X_0 that are contained in the set X. We verify if there exists a trajectory that can reach an unsafe set X_u.

For a map $f : A \to B$ and subset $C \subset A$, we write $f(C) \equiv \{f(x) |\ x \in C\}$. Thus, the safety of a system Γ is defined as follows.

Definition 114 (Safety)**.** Let $\Gamma = (f, X, X_0, X_\mathrm{u})$ be given. A trajectory $\phi_{X_0} : [0, T] \to \mathbb{R}^n$ is unsafe if there exists a time $t \in [0, T]$, such that $\phi_{X_0}([0, t]) \cap X_\mathrm{u} \neq \emptyset$ and $\phi_{X_0}([0, t]) \subseteq X$.

We say that a system Γ is safe if there are no unsafe trajectories.

To verify the safety of Γ, we use the following proposition. We do not consider the so-called strict barrier certificate in this paper, as the weak barrier certificate is more computationally tangible. However, the results presented in the paper can easily be derived for strict barrier certificates.

Proposition 45 (Weak barrier certificate [7, 8])*: Let $\Gamma = (f, X, X_0, X_\mathrm{u})$ be given. If there exists a differentiable function $B : X \to \mathbb{R}$ satisfying*

$$B(x) \leq 0 \qquad \forall x \in X_0, \tag{10.3a}$$

$$B(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{10.3b}$$

$$L_f B(x) \leq 0 \qquad \forall x \in X. \tag{10.3c}$$

Then the system Γ is safe.

Proposition 45 states that a trajectory of a system initialized in a state within the zero sublevel set of a nonincreasing function (along system trajectories), cannot reach the complement of the zero sublevel set.

Notation

For $k \in \mathbb{N}$. Given $x = (x_1, \ldots, x_k) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k}$, with $x_i \in \mathbb{R}^{n_i}$, we define $\hat{x}_i \equiv (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. Similarly, given a sequence of maps $(h_1, \ldots, h_k)$, we define $\hat{h}_i \equiv (h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_k)$.

3 Compositional Barrier Certificates

In this section, we pose the safety verification as a compositional problem, by assuming that a dynamical system is given as an interconnection of subsystems. This is based on conditions given in [12].

First, we provide the definition of an interconnected system and provide a small example, to give the necessary intuition. Note that any system (10.1) can be given as an interconnection of subsystems.

Definition 115. Let $\Gamma = (f, X, X_0, X_\mathrm{u})$ be a dynamical system with

$$\dot{x} = f(x), \tag{10.4}$$

where $x \in \mathbb{R}^n$ is the state.

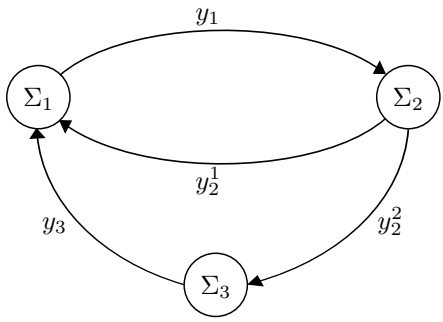


Figure 10.1: Interconnection of three subsystems $\Sigma_1, \Sigma_2, \Sigma_3$.

Let $k \in \mathbb{N}$ and $x = (x_1, \ldots, x_k)$. For $i = 1, \ldots, k$, let $x_i \in \mathbb{R}^{n_i}$, let $g_i : \mathbb{R}^{n-n_i} \to \mathbb{R}^{m_i}$ and $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$ be continuous maps, and let $q \equiv \sum_i q_i$. Let $X = X_1 \times \cdots \times X_k$, $X_0 = X_{0,1} \times \cdots \times X_{0,k}$, and $X_\mathrm{u} = X_{\mathrm{u},1} \times \cdots \times X_{\mathrm{u},k}$. We say that the system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\})$ with

$$\begin{aligned} \dot{x}_i &= f_i(x_i, g_i(\hat{x}_i)), \\ y_i &= h_i(x_i) \end{aligned} \tag{10.5}$$

where the map g_i gives the inputs to subsystem i and that the map h_i gives the outputs of subsystem i for $i = 1, \ldots, k$ is an interconnected system of $f(x)$ if

$$f(x) = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{10.6}$$

for all $x \in X$ and there exist maps $e_i : \mathbb{R}^{q-q_i} \to \mathbb{R}^{m_i}$ such that

$$g_i = e_i \circ \hat{h}_i. \tag{10.7}$$

The compositional setup is clarified by providing a system consisting of three interconnected subsystems. The interconnection of the three subsystems is shown in Figure 10.1.

Let each subsystem be described by a system of continuous ordinary differential equations and an output map

$$\Sigma_1 : \begin{cases} \dot{x}_1 = f_1(x_1, g_1(\hat{x}_1)) \\ y_1 = h_1(x_1) \end{cases} \tag{10.8a}$$

$$\Sigma_2 : \begin{cases} \dot{x}_2 = f_2(x_2, g_2(\hat{x}_2)) \\ y_2 = h_2(x_2) \end{cases} \tag{10.8b}$$

$$\Sigma_3 : \begin{cases} \dot{x}_3 = f_3(x_3, g_3(\hat{x}_3)) \\ y_3 = h_3(x_3), \end{cases} \tag{10.8c}$$

where $x_i \in X_i \subseteq \mathbb{R}^{n_i}$ is the state and $y_i \in \mathbb{R}^{q_i}$ is the output given by the map $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$. Note that the interconnection of the three subsystems is defined by $e_i : \mathbb{R}^{q-q_i} \to \mathbb{R}^{m_i}$. From Figure 10.1, it is seen that

$$e_1 : (y_2^1, y_2^2, y_3) \mapsto (y_2^1, y_3), \tag{10.9a}$$

$$e_2 : (y_1, y_3) \mapsto y_1, \tag{10.9b}$$

$$e_3 : (y_1, y_2^1, y_2^2) \mapsto y_2^2. \tag{10.9c}$$

Note that the interconnected system induces a natural graph structure, where there are no self-loops and there is only one edge from one vertex to another. The graph can be described by an adjacency matrix $E \in \mathbb{R}^k \times \mathbb{R}^k$, where $E(i, j) = 1$ if there is an edge between subsystem i and j, with the head at subsystem i and the tail at subsystem j. Note that the ith row of E can be derived from e_i. For the graph in Figure 10.1 the adjacency matrix is

$$E = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{10.10}$$

Finally, we can state the combinatorial condition for safety.

Corollary 14 ([12])*: Let $k \in \mathbb{N}$ and let the dynamical system $\Gamma = (\{f_i\}, \{X_i\}, \{X_{0,i}\}, \{X_{\mathrm{u},i}\})$ be given. If there exist differentiable functions $B_i : X_i \to \mathbb{R}$, constants $\alpha_i, \beta_i \in \mathbb{R}$, and continuous functions $\gamma_i : \mathbb{R}^{q_i+m_i} \to \mathbb{R}$ for $i = 1, \ldots, k$ such that*

$$B_i(x_i) + \alpha_i \leq 0 \qquad \forall x_i \in X_{0,i}, \tag{10.11a}$$

$$B_i(x_i) - \beta_i > 0 \qquad \forall x_i \in X_{\mathrm{u},i}, \tag{10.11b}$$

$$L_{f_i} B_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \qquad \forall (x_i, \hat{x}_i) \in X_i \times \hat{X}_i, \tag{10.11c}$$

and

$$\sum_i \alpha_i \geq 0, \sum_i \beta_i \geq 0, \sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in X_i \times \hat{X}_i. \tag{10.11d}$$

Then the system Γ is safe.

4 Existence of Compositional Barrier Certificates

In this section, we show that Proposition 45 and Corollary 14 are equivalent, if φ is assumed to be additively separable and the differential of the output maps has constant rank. First, we define an additively separable function and state a necessary assumption on the output map. Then we provide three lemmas from which the main theorem follows. Finally, we provide two examples, one of which the compositional method cannot be used to verify.

Definition 116. Let $k \in \mathbb{N}$ and $i = 1, \ldots, k$. We say that a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is additively separable in $x = (x_1, \ldots, x_k)$ if there exist functions $\varphi_i : \pi_i(\mathbb{R}^n) \to \mathbb{R}$, where π_i is a projection that takes $(x_1, \ldots, x_k)$ to x_i such that

$$\varphi(x) = \sum_i \varphi_i(x_i) \qquad \forall x \in \mathbb{R}^n, \tag{10.12}$$

where $x_i \in \mathbb{R}^{n_i}$ and $n = \sum_i n_i$.

Lemma 15 relies on the generation of a coordinate transformation that can be generated if the following assumption on the output map holds.

Assumption 2: *Let Dh_i be the differential of h_i. For $i = 1, \dots, k$*

$$Dh_i(x_i) \tag{10.13}$$

has constant rank.

The assumption guarantees that an output cannot occasionally "disappear".

A necessary and sufficient condition is given below for the composition of inequality constraints.

Lemma 12. *Let $k \in \mathbb{N}$. For $i = 1, \dots, k$, let $n_i \in \mathbb{N}$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ be a continuous function, and $X_i \subseteq \mathbb{R}^{n_i}$ be compact. There exist constants $c_i \in \mathbb{R}$ such that*

$$f_i(x_i) - c_i \leq 0 \quad \forall x_i \in X_i \text{ and} \tag{10.14a}$$

$$\sum_i c_i \leq 0 \tag{10.14b}$$

if and only if

$$\sum_i f_i(x_i) \leq 0 \quad \forall x_i \in X_i. \tag{10.15}$$

Proof. It is seen that (10.14) implies (10.15), by summing (10.14a) for $i = 1, \dots, k$

$$\sum_i f_i(x_i) \leq \sum_i c_i \quad \forall x_i \in X_i, \tag{10.16}$$

which by (10.14b) is bounded from above by zero.
To show that (10.15) implies (10.14), let

$$\bar{\Gamma} \equiv \sup\{\sum_i f_i(x_i) | x_i \in X_i\},$$

$$\bar{x} \equiv \arg\sup\{\sum_i f_i(x_i) | x_i \in X_i\},$$

$$\tilde{\Gamma} \equiv \sum_i \sup\{f_i(x_i) | x_i \in X_i\}, \text{ and}$$

$$\tilde{x} \equiv \arg \sum_i \sup\{f_i(x_i) | x_i \in X_i\}.$$

It is seen that $f_i(\bar{x}_i) \leq f_i(\tilde{x}_i)$, which implies that $\bar{\Gamma} \leq \tilde{\Gamma}$.

Note that $\tilde{x} \in X_1 \times \cdots \times X_k$, i.e., $\bar{\Gamma} \geq \tilde{\Gamma}$. This implies that $\bar{\Gamma} = \tilde{\Gamma} \leq 0$.

For $i = 1, \dots, k$, let $c_i \equiv \sup\{f_i(x_i) | x_i \in X_i\}$; then condition (10.14b) is satisfied. Furthermore, per definition of c_i

$$f_i(x_i) \leq c_i \quad \forall x_i \in X_i, \tag{10.17}$$

which implies (10.14a). □

From Lemma 12, it is seen that an inequality (10.15) in n variables is equivalent to k inequalities in n_i variables and an inequality constraint involving only constants. This is used later to decompose inequality constraints.

The following result is used in Lemma 15, to reduce the number for coupling variables in the compositional condition for safety in Corollary 14, by exploiting Assumption 2.

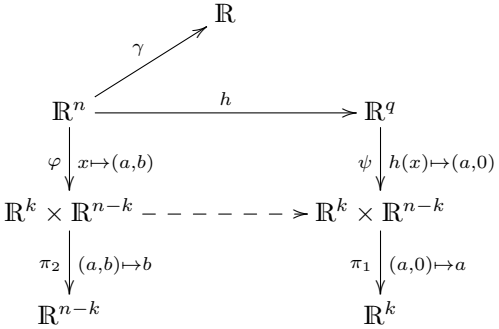
Lemma 13. *Let $\gamma : \mathbb{R}^n \to \mathbb{R}$ be a continuous function and let $h : \mathbb{R}^n \to \mathbb{R}^q$ be a smooth map such that Dh has constant rank k. Then there are smooth maps $\bar{h} : U \to Z$, with $U \subset \mathbb{R}^n$ and $Z \subset \mathbb{R}^{n-k}$, such that $D\bar{h}$ has constant rank $n-k$, and continuous functions $\tilde{\gamma} : Y \to \mathbb{R}$, with $Y \subset \mathbb{R}^{q+(n-k)}$ such that*

$$\gamma(x) = \tilde{\gamma}(h(x), \bar{h}(x)) \qquad \forall x \in \mathbb{R}^n. \tag{10.18}$$

Proof. We use Constant Rank Theorem, recalled here for completeness: Let V, W be m, n-dimensional vector spaces and $U \subset V$ an open set. If $h : U \to W$ is a smooth map such that Dh has constant rank k in U, then for each point $p \in U$ there are charts (U, φ) and (W, ψ) containing p, $h(p)$ such that

$$\psi \circ h \circ \varphi^{-1} : (x_1, \ldots, x_m) \mapsto (x_1, \ldots, x_k, 0, \ldots, 0). \tag{10.19}$$

To see how $\tilde{\gamma}$ and $\bar{h}$ can be generated, we give the following commutative diagram based on Constant Rank Theorem.



Now, the functions $\bar{h}$ and γ that satisfy (10.18) are $\bar{h} = \pi_2 \circ \varphi$, where $\pi_2 : (x_1, \ldots, x_n) \mapsto (x_{k+1}, \ldots, x_n)$ and let $\tilde{\gamma} = \gamma \circ \varphi^{-1} \circ (\pi_1 \circ \psi, id)$, where $\pi_1 : (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_k)$ and id is the identity map of dimension $n-k$.

The local patches agree on their intersection, as the function γ is defined globally. □

Lemma 14. *Let $y = (y_1, \ldots, y_k) \in Y_1 \times \cdots \times Y_k \subseteq \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_k}$, $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$, let $m = \sum_i m_i$, and $\gamma_i : \mathbb{R}^{m_i} \times \mathbb{R}^{n_i} \times \mathbb{R}^{m-m_i} \to \mathbb{R}$. Then for every $(y_1, \ldots, y_k)$*

$$\max_{(z_1,\ldots,z_k) \in h_1^{-1}(y_1) \times \cdots \times h_k^{-1}(y_k)} \sum_i \gamma_i(y_i, z_i, \hat{y}_i) = \sum_i \max_{z_i \in h_i^{-1}(y_i)} \gamma_i(y_i, z_i, \hat{y}_i). \tag{10.20}$$

Proof. For any y, let

$$\begin{aligned}
\bar{\Gamma}(y) &\equiv \max_{(z_1,\ldots,z_k)\in h_1^{-1}(y_1)\times\cdots\times h_k^{-1}(y_k)} \sum_i \gamma_i(y_i, z_i, \hat{y}_i), \\
\bar{z}(y) &\equiv \arg \max_{(z_1,\ldots,z_k)\in h_1^{-1}(y_1)\times\cdots\times h_k^{-1}(y_k)} \sum_i \gamma_i(y_i, z_i, \hat{y}_i), \\
\tilde{\Gamma}(y) &\equiv \sum_i \max_{z_i\in h_i^{-1}(y_i)} \gamma_i(y_i, z_i, \hat{y}_i), \text{ and} \\
\tilde{z}(y) &\equiv \arg \sum_i \max_{z_i\in h_i^{-1}(y_i)} \gamma_i(y_i, z_i, \hat{y}_i).
\end{aligned}$$

It is seen that $\gamma_i(y_i, \bar{z}_i(y), \hat{y}_i) \leq \gamma_i(y_i, \tilde{z}_i(y), \hat{y}_i)$; hence, $\bar{\Gamma}(y) \leq \tilde{\Gamma}(y)$.

For any y, $\tilde{z}(y) \in h_1^{-1}(y_1) \times \cdots \times h_k^{-1}(y_k)$. This implies $\bar{\Gamma}(y) \geq \tilde{\Gamma}(y)$; hence, $\bar{\Gamma}(y) = \tilde{\Gamma}(y)$. Note that this is a consequence of γ_i's independence of $\hat{z}_i$. □

The final lemma on the decomposition of inequality constraints is shown next.

Lemma 15. *Let $k \in \mathbb{N}$. For $i \in \{1, \ldots, k\}$, let*

- *$m_i, n_i, q_i \in \mathbb{N}$ and define $q \equiv \sum_i q_i$,*
- *$V_i = X_i \times \hat{X}_i \subseteq \mathbb{R}^{n_i} \times \mathbb{R}^{n-n_i}$ be compact and be the closure of an open set,*
- *$f_i : \mathbb{R}^{n_i+m_i} \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R}^{n-n_i} \to \mathbb{R}^{m_i}$ be continuous maps, defined in* (10.6) *and* (10.7)*,*
- *$\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ be a continuous function,*
- *$h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{q_i}$ be a smooth map such that Dh_i has constant rank r_i.*

There exist continuous functions $\gamma_i : \mathbb{R}^{q_i+m_i} \to \mathbb{R}$ such that for all $(x_i, \hat{x}_i) \in V_i$

$$\varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)), \text{ and} \tag{10.21a}$$

$$\sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \tag{10.21b}$$

if and only if

$$\sum_i \varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in V_i. \tag{10.22}$$

Proof. It is seen that (10.21) implies (10.22), by summing (10.21a) for $i = 1, \ldots, k$, such that

$$\sum_i \varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \leq \sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \qquad \forall (x_i, \hat{x}_i) \in V_i, \tag{10.23}$$

which by (10.21b) is bounded from above by zero.

To show that (10.22) implies (10.21), let

$$\bar{\gamma}_i(x_i, g_i(\hat{x}_i)) \equiv \varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) \;\; \forall (x_i, \hat{x}_i) \in V_i. \tag{10.24}$$

By (10.22)

$$\sum_i \bar{\gamma}_i(x_i, g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in V_i. \tag{10.25}$$

Lemma 13 states that by assuming that Dh_i has constant rank r_i; there exist functions $\tilde{\gamma}_i : \mathbb{R}^{q_i+(n_i-r_i)+m_i} \to \mathbb{R}$ and maps $\bar{h}_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i-r_i}$ such that for all $(x_i, \hat{x}_i) \in V_i$

$$\tilde{\gamma}_i(h_i(x_i), \bar{h}_i(x_i), g_i(\hat{x}_i)) = \bar{\gamma}_i(x_i, g_i(\hat{x}_i)). \tag{10.26}$$

We rewrite (10.25) as follows

$$\sum_i \tilde{\gamma}_i(h_i(x_i), \bar{h}_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in V_i. \tag{10.27}$$

Recall that $g_i = e_i \circ \hat{h}_i$. For every $(y_1, \ldots, y_k)$, the following equality holds

$$\begin{aligned} &\max_{(z_1,\ldots,z_k) \in h_1^{-1}(y_1) \times \cdots \times h_k^{-1}(y_k)} \sum_i \tilde{\gamma}_i(y_i, \bar{h}_i(z_i), e_i(\hat{y}_i)) \\ &\qquad = \sum_i \max_{z_i \in h_i^{-1}(y_i)} \tilde{\gamma}_i(y_i, \bar{h}_i(z_i), e_i(\hat{y}_i)), \end{aligned} \tag{10.28}$$

since the choice of z_i is independent of the choice of z_j for all $i \neq j$. This is seen from Lemma 14.

We define

$$\gamma_i(y_i, e_i(\hat{y}_i)) \equiv \max_{z_i \in h_i^{-1}(y_i) \cap X_i} \tilde{\gamma}_i(y_i, \bar{h}_i(z_i), e_i(\hat{y}_i)). \tag{10.29}$$

It is seen from (10.27) that

$$\sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall (x_i, \hat{x}_i) \in V_i. \tag{10.30}$$

Furthermore, by (10.24) for all $(x_i, \hat{x}_i) \in V_i$

$$\varphi_i(x_i) f_i(x_i, g_i(\hat{x}_i)) = \bar{\gamma}_i(x_i, g_i(\hat{x}_i)) = \tilde{\gamma}_i(h_i(x_i), \bar{h}_i(x_i), g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \tag{10.31}$$

We show that γ_i is continuous. According to Theorem 4 and Theorem 5 in [13], γ_i is continuous at y_i if $\tilde{\gamma}_i$ is continuous and $h_i^{-1} \cap X_i$ is continuous as a multivalued map and $h_i^{-1}(y_i) \cap X_i$ is compact. In the following, we discard the indices on h and denote X_i by V.

The map h^{-1} is continuous, as h is a composition of a projection p with homeomorphisms ψ and φ, since h according to Lemma 13 can be written

$$h = \psi^{-1} \circ \psi \circ h \circ \varphi^{-1} \circ \varphi,$$

where $p = \psi \circ h \circ \varphi^{-1}$ is a projection. Therefore, h is also an open map.

$h^{-1}(x) \cap V$ is upper semi-continuous at x_0 if for any open set $U \supset h^{-1}(x_0) \cap V$ there exists an open set W of x_0 such that $h^{-1}(W) \cap V \subset U$. Since h^{-1} is continuous,

it is known that for any open set $\bar{U} \supset h^{-1}(x_0)$ there exists an open set $\bar{W}$ such that $h^{-1}(\bar{W}) \subset \bar{U}$. Therefore, $h^{-1}(\bar{W}) \cap V \subset \bar{U} \cap V = U$.

$h^{-1}(x) \cap V$ is lower semi-continuous at x_0 if for any $y_0 \in h^{-1}(x_0) \cap V$ and for any open neighborhood $\Omega(y_0)$ of y_0, there exists open neighborhood $\Omega(x_0)$ of x_0 such that for any $x \in \Omega(x_0)$,

$$h^{-1}(x) \cap V \cap \Omega(y_0) \neq \emptyset.$$

From the continuity of h^{-1}, we know that for any open neighborhood $\Omega(y_0)$ there exists open neighborhood $\bar{\Omega}(x_0)$ such that for any $x \in \bar{\Omega}(x_0)$

$$h^{-1}(x) \cap \Omega(y_0) \neq \emptyset.$$

By assumption, $y_0 \in h^{-1}(x_0) \cap V$ and $y_0 \in \Omega(y_0)$. For any $y_0 \in V$ (also $y_0 \in \mathrm{Bd}(V)$), $\Omega(y_0) \cap \mathrm{int}(V)$ is an open and nonempty set. Therefore, the set

$$B \equiv h^{-1}(\bar{\Omega}(x_0)) \cap \mathrm{int}(V) \cap \Omega(y_0)$$

is open.

Since h is an open map, $h(B)$ is an open neighborhood of x_0. Let $\tilde{\Omega}(x_0) = h(B)$, then for every $x \in \tilde{\Omega}(x_0)$, $h^{-1}(x) \cap B \neq \emptyset$. This implies that for every $x \in \tilde{\Omega}(x_0)$

$$h^{-1}(x) \cap \mathrm{int}(V) \cap \Omega(y_0) \neq \emptyset.$$

This implies that $h^{-1}(x) \cap V$ is continuous.

For every x, the sets $h^{-1}(x)$ is closed and V is compact. Hence, $h^{-1}(x) \cap V$ is compact. This proves the continuity of γ.

□

Notice the importance of using γ_i in oppose to $\bar{\gamma}_i$. γ_i is only a function of the outputs of subsystem i, while $\bar{\gamma}_i$ is a function of its entire state vector. This implies that the dimension of the coupling is drastically reduced if the number of output variables is small compared to the number of states. Furthermore, it is important to note that γ_i is continuous, as it enables γ_i to be approximated arbitrarily close by polynomials on a compact set. This is favorable, as polynomial inequality and equality constraints can be solved algorithmically by use of sum of squares programming [14].

We can now state when Proposition 45 and Corollary 14 are equivalent.

Theorem 18*: Let $k \in \mathbb{N}$, and let*

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{10.32}$$

be an interconnected system of $f(x)$.

There exists an additively separable continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that

$$\varphi(x) \leq 0 \qquad \forall x \in X_0, \tag{10.33a}$$

$$\varphi(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{10.33b}$$

$$L_f \varphi(x) \leq 0 \qquad \forall x \in X \tag{10.33c}$$

if and only if for $i = 1, \ldots, k$ *there exist continuous functions* $\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ *and* $\gamma_i : \mathbb{R}^{q_i+m_i} \to \mathbb{R}$ *and constants* $\alpha_i, \beta_i \in \mathbb{R}$ *such that*

$$\varphi_i(x_i) + \alpha_i \leq 0 \qquad \forall x \in X_0, \tag{10.34a}$$
$$\varphi_i(x_i) - \beta_i > 0 \qquad \forall x \in X_{\mathrm{u}}, \tag{10.34b}$$
$$L_{f_i}\varphi_i(x_i, g_i(\hat{x}_i)) \leq \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \ \ \forall x \in X \tag{10.34c}$$

and

$$\sum_i \alpha_i \geq 0, \tag{10.34d}$$
$$\sum_i \beta_i \geq 0, \ \textit{and} \tag{10.34e}$$
$$\sum_i \gamma_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0. \tag{10.34f}$$

Proof. The conditions (10.34) implies (10.33) directly. Therefore, we only show the opposite direction. Suppose φ is additively separable, then per definition there exist continuous functions $\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ such that

$$\varphi(x) = \sum_i \varphi_i(x_i) \qquad \forall x \in X. \tag{10.35}$$

This implies that (10.33) can be written as

$$\sum_i \varphi_i(x_i) \leq 0 \qquad \forall x \in X_0, \tag{10.36a}$$
$$\sum_i \varphi_i(x_i) > 0 \qquad \forall x \in X_{\mathrm{u}}, \text{ and} \tag{10.36b}$$
$$\sum_i L_{f_i}\varphi_i(x_i, g_i(\hat{x}_i)) \leq 0 \qquad \forall x \in X. \tag{10.36c}$$

From Lemma 12, it follows directly that (10.36a) is equivalent to (10.34a) and (10.34d). In addition, (10.36b) is equivalent to (10.34b) and (10.34e). Finally, Lemma 15 shows that (10.33c) is equivalent to (10.34c) and (10.34f). □

Remark that to generate the additively separable barrier functions, one should only use the sum of bases in x_i for $i \in \{1, \ldots, k\}$.

To clarify the theorem, Example 17 demonstrates the necessity of α_i and β_i.

Example 17. Consider the following simple dynamical system

$$\Sigma_1 : \dot{x} = -x \tag{10.37a}$$
$$\Sigma_2 : \dot{y} = -y \tag{10.37b}$$

where $x, y \in \mathbb{R}$. The system is split into two independent dynamical systems Σ_1 and Σ_2. The set of initial states is $X_0 = [4, 5] \times [4, 5]$ and the set of unsafe states is $X_{\mathrm{u}} = [1, 2] \times [4, 5]$. The vector field, X_0, and X_{u} are illustrated in Fig. 10.2.

From X_0 and X_{u}, it is seen that there exists no function φ_2 such that $\varphi_2(y) \leq 0$ for all $y \in [4, 5]$ and $\varphi_2(y) > 0$ for all $y \in [4, 5]$. Therefore, the constants α_i and β_i are

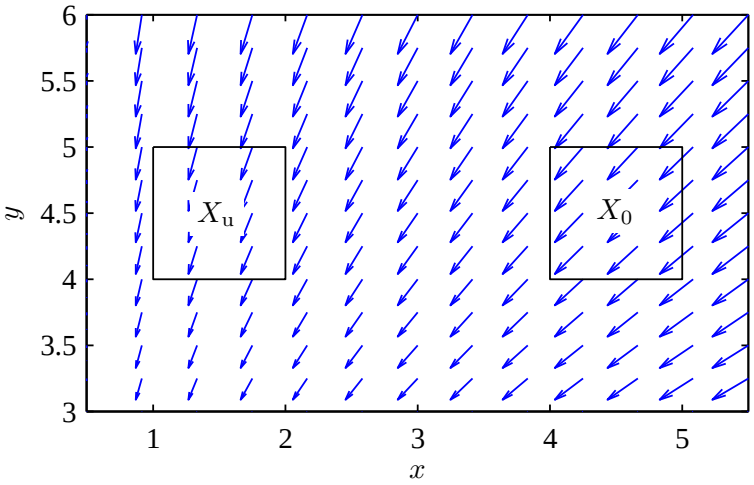


Figure 10.2: Vector field (blue arrows) and safe and unsafe sets (black boxes).

necessarily different from zero, even though the dynamics is completely decoupled for the two subsystems.

An additively separable barrier certificate $\varphi = \sum_i \varphi_i$ is given by $\varphi_1 = 0.26x^4 - 2x^3 + 3x^2 + 2.65$ and $\varphi_2 = 0.1y^2 - 1.6$. For $\alpha_1 = 2.2$, $\alpha_2 = -1.8$, $\beta_1 = 1.2$, $\beta_2 = -0.8$, the conditions in (10.34) are satisfied.

Note other compositional methods for the analysis of dynamical systems, such as [10], suffer similar deficits due to the restriction to additively separable functions.

5 Refined Compositional Analysis

We propose another condition for safety, which at the cost of more coupling variables handles more general problems. The idea is to let each φ_i depend on both x_i and $g_i(\hat{x}_i)$.

To simplify the notation of the problem, we define the set of neighbors for subsystem i, as the set of subsystems, which has an output that is an input to subsystem i. The set of neighbors is defined from the adjacency matrix E, see (10.10), describing the interconnection of the subsystems. We say that the neighbors of subsystem i have the following indices

$$\mathcal{N}_i = \{j \in \{1, \ldots, k\} | E(i, j) = 1\}. \tag{10.38}$$

We define $\bar{\mathcal{N}}_i \equiv \mathcal{N}_i \cup \{i\}$. The complement of $\bar{\mathcal{N}}_i$ is given as

$$\bar{\mathcal{N}}_i^{\mathrm{c}} = \{1, \ldots, k\} \backslash \bar{\mathcal{N}}_i. \tag{10.39}$$

Let $z = (z_1, \ldots, z_k)$ and $A \subseteq \{1, \ldots, k\}$, then we define $\hat{z}_A \equiv \{z_i | i \in \{1, \ldots, k\} \backslash A\}$ and $z_A \equiv \sum_{i \in A} z_i$.

Now, we can state the refined safety condition as follows.

Proposition 46: *Let* $k \in \mathbb{N}$, *and let*

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} f_1(x_1, g_1(\hat{x}_1)) \\ \vdots \\ f_i(x_i, g_i(\hat{x}_i)) \\ \vdots \\ f_k(x_k, g_k(\hat{x}_k)) \end{bmatrix} \tag{10.40}$$

be an interconnected system of $f(x)$.

There exists a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ *given by*

$$\varphi(x) = \sum_i \varphi_i(x_i, g_i(\hat{x}_i)) \qquad \forall x \in \mathbb{R}^n \tag{10.41}$$

such that

$$\varphi(x) \leq 0 \qquad \forall x \in X_0, \tag{10.42a}$$
$$\varphi(x) > 0 \qquad \forall x \in X_\mathrm{u}, \text{ and} \tag{10.42b}$$
$$L_f\varphi(x) \leq 0 \qquad \forall x \in X \tag{10.42c}$$

if and only if for $i \in \{1, \ldots, k\}$ *there exist continuous functions* $\gamma_i : \mathbb{R}^{q_{\bar{\mathcal{N}}_i} + m_{\mathcal{N}_i}} \to \mathbb{R}$, $\alpha_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$, *and* $\beta_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$ *such that*

$$\varphi_i(x_i, g_i(\hat{x}_i)) + \alpha_i(h_i(x_i), g_i(\hat{x}_i)) \leq 0 \qquad \forall x \in X_0, \tag{10.43a}$$
$$\varphi_i(x_i, g_i(\hat{x}_i)) - \beta_i(h_i(x_i), g_i(\hat{x}_i)) > 0 \qquad \forall x \in X_\mathrm{u}, \tag{10.43b}$$
$$\sum_{j \in \bar{\mathcal{N}}_i} \frac{\partial \varphi_i}{\partial x_j}(x_i, g_i(\hat{x}_i)) f_j(x_j, g_j(\hat{x}_j)) \leq \gamma_i(\hat{x}_{\mathcal{N}_i^\mathrm{c}}, \hat{g}_{\mathcal{N}_i^\mathrm{c}}(\hat{x}_{\mathcal{N}_i^\mathrm{c}})) \ \forall x \in X \tag{10.43c}$$

and

$$\begin{aligned} \sum_i \alpha_i(h_i(x_i), g_i(\hat{x}_i)) &\geq 0, \\ \sum_i \beta_i(h_i(x_i), g_i(\hat{x}_i)) &\geq 0, \\ \sum_i \gamma_i(\hat{x}_{\mathcal{N}_i^\mathrm{c}}, \hat{g}_{\mathcal{N}_i^\mathrm{c}}(\hat{x}_{\mathcal{N}_i^\mathrm{c}})) &\leq 0. \end{aligned} \tag{10.43d}$$

Proof. The equivalence between (10.42a) and (10.43a), and (10.42b) and (10.43b) follows directly from the proof of Lemma 15 starting from (10.27) to the end of the proof. To obtain (10.43c), observe that $\frac{\partial \varphi_i}{\partial x_j}(x_i, g_i(\hat{x}_i))$ is only nonzero for $j \in \bar{\mathcal{N}}_i$. For each nonzero partial derivative, $\partial \varphi_i / \partial x_j$ is multiplied by $f_j(x_j, g_j(\hat{x}_j))$ that is a function of x_j and $g_j(\hat{x}_j)$. This implies that the left hand side of (10.43c) depends on $\hat{x}_{\bar{\mathcal{N}}_i^\mathrm{c}}$ (states of subsystem i and its neighbors) and $\hat{g}_{\mathcal{N}_i^\mathrm{c}}(\hat{x}_{\mathcal{N}_i^\mathrm{c}})$ (the neighbors inputs - not its own, as their states are already in γ_i). □

The seemingly subtle change of φ_i has a great impact on the number of coupling variables involved in the generation of the barrier certificate. Therefore, one should only include $g_i(\hat{x}_i)$ in φ_i if it is really necessary. Remark that a subset of functions φ_i may be dependent of $g_i(\hat{x}_i)$, while others may only depend on x_i.

6 Conclusion

We have classified the barrier certificates, which can be generated by a proposed compositional method for verifying the safety of continuous dynamical systems. It is shown that even for some linear systems, the compositional method fails to verify the safety.

Even though the compositional method is not as general as a centralized safety verification, it is very useful in the verification of high-dimensional systems, since it scales well in the number of states in the system. Therefore, counterexamples where the compositional method does not apply should be used to generate "good" decompositions of systems.

A second compositional condition for safety was proposed, which alleviates some of the issues of the initial method, but has a higher computational cost. Therefore, the choice of method is a compromise between generality and computational complexity. Therefore, our future work is to identify the necessary structure of the barrier certificate based on the vector field.

References

[1] H. Guéguen, M.-A. Lefebvre, J. Zaytoon, and O. Nasri, "Safety verification and reachability analysis for hybrid systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 25–36, 2009.

[2] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[3] J. Ding, J. H. Gillula, H. Huang, M. P. Vitus, W. Zhang, and C. J. Tomlin, "Hybrid systems in robotics," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 33–43, September 2011.

[4] A. Girard and G. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3927, pp. 272–286.

[5] A. Abate, A. Tiwari, and S. Sastry, "Box invariance in biologically-inspired dynamical systems," *Automatica*, vol. 45, no. 7, pp. 1601–1610, 2009.

[6] C. Sloth and R. Wisniewski, "Verification of continuous dynamical systems by timed automata," *Formal Methods in System Design*, vol. 39, no. 1, pp. 47–82, 2011.

[7] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, August 2007.

[8] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 2993, pp. 271–274.

[9] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "SOSTOOLS and its control applications," in *Positive Polynomials in Control*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 2005, vol. 312, pp. 273–292.

[10] U. Topcu, A. Packard, and R. Murray, "Compositional stability analysis based on dual decomposition," in *Proceedings of the 48th IEEE Conference on Decision and Control*, December 2009, pp. 1175–1180.

[11] F. Kerber and A. van der Schaft, "Compositional analysis for linear control systems," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. New York, NY, USA: ACM, 2010, pp. 21–30.

[12] C. Sloth, G. J. Pappas, and R. Wisniewski, "Compositional safety analysis using barrier certificates," in *Proceedings of Hybrid Systems: Computation and Control*, 2012, pp. 15–23.

[13] J.-P. Aubin and A. Cellina, *Differential Inclusions*. Springer-Verlag, 1984.

[14] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. 293–320, 2003.

Paper F

Towards Safe Robotic Surgical Systems

Christoffer Sloth, Rafael Wisniewski, Jesper A. Larsen, John Leth, and Johan Poulsen

This paper is submitted to:
IEEE Transactions on Biomedical Engineering.

The paper is extended with a description of the modeling in an appendix.

Abstract

A proof of safety is paramount for an autonomous robotic surgical system to ensure that it does not cause trauma to patients. However, a proof of safety is rarely constructed, as surgical systems are too complex to be dealt with by most formal verification methods.

In this paper, we design a controller for motion compensation in beating-heart surgery, and prove that it is safe, i.e., the surgical tool is kept within an allowable distance and orientation of the heart. We solve the problem by simultaneously finding a control law and a barrier function using a sequential method.

The motion compensation system is simulated from several initial conditions to demonstrate that the designed control system is safe for every admissible initial condition.

1 Introduction

In the past decades, there has been an immense development of robotic surgical systems. This has led to regular use of robotic assistance in surgery of human patients.

In cardiac bypass surgery several advantages exist if the procedure can be carried out without the use of extracorporeal circulation. However, this means that the delicate coronary artery anastomosis has to be performed on a beating heart. This is a very technical demanding task due to the pulsating movement of the heart.

It was proposed in [1] to provide a video image in the robotic console to the surgeon that looks as if the heart is static (although it still is beating), and move the surgical instruments in a similar pulsating motion as the heart is beating. Then the surgeon can perform the procedure using a real time image where the heart as well as the surgical instruments looks static. This makes subsequent coronary artery anastomosis much easier for the surgeon to do.

In this paper, we focus on the design of the required control system for moving the surgical instruments. The addition of automatic control introduces a risk for the surgical robot to cause trauma to patients, without human intervention [2], as experienced with the Therac-25 [3]. To alleviate this potential issue, the safety (correct behavior) of control systems must be formally proved before they are employed in surgical systems.

In the past two decades, lots of methods for designing safe dynamical systems have been developed [4]. However, these methods have not been frequently applied to robotic surgical systems. An exception is [5] that presents a verification of high level plans for composing procedures in robotic surgery in relation to puncturing. Formal verification has not yet been used much in robotic surgery, since methods based on reachability can only handle systems with a few continuous states [4], and other more scalable methods do not apply to the nonlinear dynamics of the robotic system. Recently, attention has been directed towards the development of methods for improving the scalability of verification methods by using assume-guarantee reasoning [6] and compositional computational techniques [7].

To design a safe motion compensation control system for beating-heart surgery, we use the barrier certificate method, which was developed for safety verification [8], and extended to the design of safe controllers [9]. We use an experimentally generated model

from [10] to describe the movement of the heart as a combination of respiratory motion and heart beating. The safe control system is designed for a 6 degree of freedom robot based on the end-effector [11]. To make the controller design method applicable to high dimensional systems, we exploit ideas from the computational techniques in [6, 7] to allow a sequential calculation of controllers for each joint of the robot. This is possible since the safety requirement is decomposed into separate requirements for the tracking of each joint angle. To find the control laws, we derive a method for realizing the control law similar to [12, 9]. To delimit the content of this paper, we abstract the important image processing away and assume that the position and orientation of the heart is measured [13]. To make the study realistic, we introduce uncertainties in the model. A detailed description of the utilized models is provided in [14].

This paper is organized as follows. Section 2 presents a model of the considered system and states the addressed problem, and Section 3 presents the designed control system, where the safety of the system is guaranteed by a barrier certificate. Section 4 shows simulated trajectories satisfying the specification, and Section 5 comprises conclusions.

2 Problem Formulation

The purpose of this section is to present a mathematical model of the interrelation between the beating heart and the robotic arm that compensates for the movement of the heart. Additionally, we define safety and finalize by stating the objective of the paper in Problem 10.

Model of Beating Heart

It is chosen to base this work on an existing model of a beating heart presented in [10] that describes the movement of the coronary arteries. This model has been experimentally generated. The principle of the model is to have two independent motions: respiratory motion and beating of the heart. Both motions are combinations of rotation and translation, and as the diaphragm moves the heart, the resulting movement of a point on the heart surface is a combination of both movements. Figure 11.1 illustrates the arrangement of reference frames used in the modeling.

The objective of this paper is to design a controller that can control a surgical tool to a point of interest on the heart surface at a given orientation. To accomplish this, we model the physiological motion using time-varying coordinate transformations [15].

For a point $p \in \mathbb{R}^3$ and a coordinate frame Ψ_A, let ${}^A p$ denote the coordinates of p in the coordinate frame Ψ_A. For coordinate frame Ψ_A and Ψ_B, the coordinates ${}^A p$ and ${}^B p$ of the point p are related by

$$\begin{bmatrix} {}^A p \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^A_B R & {}^A_B p \\ 0 & 1 \end{bmatrix}}_{{}^A_B T} \begin{bmatrix} {}^B p \\ 1 \end{bmatrix}, \tag{11.1}$$

where ${}^A_B R$ is a 3×3 rotation matrix and ${}^A_B p$ is a 3×1 vector translating the origin of coordinate frame Ψ_A to the origin of coordinate frame Ψ_B.

We use this formalism to describe the behavior of the diaphragm and the heart. It is chosen to let Ψ_0 and Ψ_d coincide at time 0, i.e., ${}^0_\mathrm{d} R(0) = I$ and ${}^0_\mathrm{d} p = 0$. The diaphragm

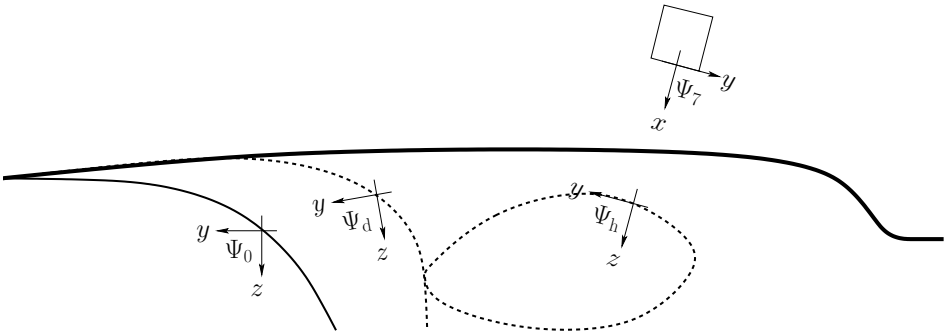


Figure 11.1: Out of scale illustration of coordinate frame in the y, z-plane. The reference frame is denoted by Ψ_0, the frame on the diaphragm is denoted by Ψ_d, the frame on the heart is denoted by Ψ_h, and the frame on the surgical tool is denoted by Ψ_7.

(Ψ_d) rotates approximately 2° around the x- and z-axis of Ψ_0 in Figure 11.1 and is translated along the y- and z-axis of Ψ_0. The behavior is periodic with a time period of 4 s. The point of interest on the heart surface (Ψ_h) rotates around the x- and y-axis of Ψ_d and is translated along the x- and y-axis of Ψ_d. This behavior has a period of 1.1 s. The resulting movement of the point of interest on the heart surface is described in the reference frame Ψ_0 using the following time-dependent transformation

$$ {}^{0}_{\mathrm{h}}T(t) = {}^{0}_{\mathrm{d}}T(t)\,{}^{\mathrm{d}}_{\mathrm{h}}T(t). \tag{11.2} $$

A patient's heart movement is nondeterministic; hence, we add a disturbance to the system that represents a variation in the time period of the heart and diaphragm.

Figure 11.2 shows the translation of the heart with respect to the reference frame Ψ_0. The elements of the transformation matrices are given by the solution of the system of

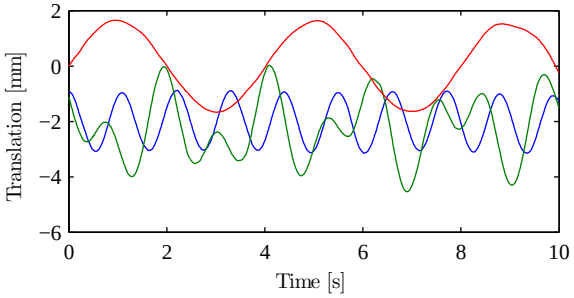


Figure 11.2: Translation of the point of interest on the heart surface along the x-axis (blue), y-axis (green), and z-axis (red) of the reference frame Ψ_0.

polynomial differential equations describing the heart model [14].

Model of Robotic Arm

We consider a 6 degree of freedom robotic arm composed of a da Vinci Surgical end-effector [11], and a 3 degree of freedom prismatic base robot. The robotic arm is described using the same formalism as the heart, but as the length of the links of the robot are fixed, only the rotations can vary. A kinematic model of the robotic arm is detailed in [14]. The angles of the 6 joints must be controlled to steer the tool frame to the point of interest on the heart surface. Therefore, we need an inverse kinematic description of the robot that, for a desired position and orientation of the tool frame, provides the six angles that realize the desired pose. The inverse kinematic description is provided in [14].

The model of the robot can be augmented with the heart model to form a system of ordinary differential equations (affine in control and disturbance)

$$\dot{x} = f(x) + g(x)u + h(x)d, \tag{11.3}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $d \in D \subseteq \mathbb{R}^p$ is the disturbance input. In this work, we control the angular velocities of the robotic arm; hence, $\dot{\theta}_i = u_i$, where u_i is the control input.

Safety of the System

We consider a control system given by $\Gamma = (f, g, h, X, X_0, X_\mathrm{u}, D)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, and $h : \mathbb{R}^n \to \mathbb{R}^{n\times p}$ are continuous, $X \subseteq \mathbb{R}^n$, $X_0 \subseteq X$, $X_\mathrm{u} \subseteq X$, and $D \subseteq \mathbb{R}^p$ is convex. The system is controlled via the continuous map $k : \mathbb{R}^n \to \mathbb{R}^m$ defining the closed-loop behavior

$$f_\mathrm{cl} : x \mapsto f(x) + g(x)k(x). \tag{11.4}$$

The closed-loop system is denoted by $\Gamma_\mathrm{cl} = (f_\mathrm{cl}, h, X, X_0, X_\mathrm{u}, D)$. For a measurable and essentially bounded disturbance function $\bar{d} : \mathbb{R}_{\geq 0} \to D$, we denote the solution of the Cauchy problem for the closed-loop system with $x(0) = x_0$ on an interval $[0, T]$ by $\phi^{\bar{d}}_{x_0}$, i.e.,

$$\frac{d\phi^{\bar{d}}_{x_0}(t)}{dt} = f_\mathrm{cl}\left(\phi^{\bar{d}}_{x_0}(t)\right) + h\left(\phi^{\bar{d}}_{x_0}(t)\right)\bar{d}(t) \tag{11.5}$$

for almost all $t \in [0, T]$. We denote the set of solutions from all initial conditions x_0 in X_0 by $\phi^{\bar{d}}_{X_0}$.

In the safety verification, we only consider trajectories initialized in X_0 that are contained in the set X. We verify if there exists a trajectory that can reach an unsafe set X_u. The safety of a system Γ_cl is defined as.

Definition 117 (Safety)**.** Let $\Gamma_\mathrm{cl} = (f_\mathrm{cl}, h, X, X_0, X_\mathrm{u}, D)$ be a control system. A trajectory $\phi^{\bar{d}}_{X_0} : [0, T] \to \mathbb{R}^n$ with disturbance $\bar{d}$ is unsafe if there exists a time $t \in [0, T]$, such that $\phi^{\bar{d}}_{X_0}([0, t]) \cap X_\mathrm{u} \neq \emptyset$ and $\phi^{\bar{d}}_{X_0}([0, t]) \subseteq X$.

We say that the system Γ_cl is safe if there are no unsafe trajectories.

The surgical robotic system is safe if the relative rotation and position between the tool frame and the heart frame is kept within certain bounds.

The objective of this work is to solve the following problem.

Problem 10*: Given a system* $\Gamma = (f, g, h, X, X_0, X_u, D)$*, design a control law* $k : \mathbb{R}^n \to \mathbb{R}^m$ *such that the system* $\Gamma_{cl} = (f_{cl}, h, X, X_0, X_u, D)$*, where* f_{cl} *is given by* (11.4)*, is safe.*

3 Control Algorithm

The purpose of this section is to present the method used for designing the safe control system. To ease the controller design, we initially decompose the specification into subspecifications.

To allow the design of a safe control system for a complex system, it is appropriate to decompose the specification into a specification for each subsystem. This enables the design of separate controllers for each subsystem, based on the assumption that each of the controllers comply with their partial specifications [6]. We use this principle in the design of the controller for the surgical robot. The requirement to the tracking of the position and orientation of the surgical tool is decomposed into requirements on the tracking of joint angles of the robot. The decomposed requirements are $|\theta_i - \theta_{i,\text{ref}}| \leq \theta_{i,\text{e}}$, where $\theta_{i,\text{e}}$ is the maximum allowed tracking error of the ith joint angle.

To satisfy the specification, we design a control system consisting of six controllers, one per joint of the robot. The barrier certificate method used for designing the controller is similar to [9]; however, we define a control barrier function in a less restrictive manner. Furthermore, we allow unknown but bounded disturbance inputs. The addition of disturbances is crucial in medical applications, where the system (the patient) is described by an uncertain model.

In the definition of a control barrier function, we denote the complement of X by X^c, and use the notion of contingent cone [16]. Let K be a nonempty subset of a space X and let x belong to K. The contingent cone to K at x is the set

$$T_K(x) = \left\{ v \in X | \lim_{h \to 0+} \inf \frac{d_K(x + hv)}{h} = 0 \right\}, \tag{11.6}$$

where $d_K(y)$ denotes the distance of y to K, defined by

$$d_K = \inf_{z \in K} ||y - z||. \tag{11.7}$$

A control barrier function is defined as

Definition 118. Given a control system Γ. A continuously differentiable function $B : X \to \mathbb{R}$ satisfying

$$X_0 \subset B^{-1}((-\infty, 0]) \subset X_u^c, \text{ and} \tag{11.8a}$$

there exists $u \in \mathbb{R}^m$ such that for any $d \in D$ and $x \in B^{-1}(0)$

$$f(x) + g(x)u + h(x)d \in T_{B^{-1}((-\infty,0])}(x) \tag{11.8b}$$

is called a control barrier function.

The intuition of a control barrier function B is to separate the initial set X_0 and the unsafe set X_u by the zero level set of B. This is ensured by (11.8a). On the zero level set, it must be possible to control the vector field to point into the zero sublevel set; hence, all

solutions avoid the unsafe set X_u that is in the complement of the zero sublevel set. This is ensured by (11.8b).

Given a control barrier function, a control law must be found that ensures the safety of the system. A selection of such a control law is provided in the following, inspired by [12]. We denote the Lie derivative of B along f by $L_f B$.

Proposition 47: Let Γ be a control system, let B be an associated proper control barrier function, and let $L_g B(x) \neq 0$ for $x \in B^{-1}(0)$. There exists a pair of real numbers (γ_1, γ_2) with $0 < \gamma_1 < \gamma_2$ such that the control

$$k = -\xi(||b||)\frac{a + \alpha + \sqrt{(a+\alpha)^2 + \kappa^2 b^{\mathrm{T}} b}}{b^{\mathrm{T}} b} b, \tag{11.9}$$

where $a \equiv L_f B$, $b^{\mathrm{T}} \equiv L_g B$, $c^{\mathrm{T}} \equiv L_h B$, $\kappa > 0$,

$$\alpha(x) \equiv \sup_{d \in D} c^{\mathrm{T}}(x) d, \tag{11.10}$$

and $\xi : \mathbb{R} \to [0, 1]$ defined by

$$\omega(z) \equiv \begin{cases} 0 & \text{if } z \leq 0 \\ \exp(-1/z) & \text{if } z > 0 \end{cases} \tag{11.11}$$

$$\xi(z) \equiv \xi_{(\gamma_1,\gamma_2)}(z) \equiv \frac{\omega(z - \gamma_1)}{\omega(z - \gamma_1) + \omega(\gamma_2 - z)}, \tag{11.12}$$

is continuous and ensures safety for the closed-loop system Γ_{cl}.

Proof. The safety of Γ_{cl} is proved by showing that the Lie derivative of the control barrier function is negative for all $x \in B^{-1}(0)$; hence, by Nagumo's Theorem $B^{-1}((-\infty, 0])$ is positively invariant [16]. The Lie derivative of the control barrier function for the closed-loop system is

$$L_{f_{\mathrm{cl}}} B = L_f B + L_g B k + L_h B d \tag{11.13}$$

$$= -\xi(||b||)(\alpha + \sqrt{(a+\alpha)^2 + \kappa^2 b^{\mathrm{T}} b}) + c^{\mathrm{T}} d. \tag{11.14}$$

The function B is proper, thus $B^{-1}(0)$ is compact. By compactness of $B^{-1}(0)$ and since $L_g B(x) \neq 0$ for $x \in B^{-1}(0)$, there exists (γ_1, γ_2) such that $\xi(||b(x)||) = 1$ for all $x \in B^{-1}(0)$ (ξ is a bump function [17]). It is seen that $-\sqrt{(a+\alpha)^2 + \kappa^2 b^{\mathrm{T}} b} < 0$, as $\kappa^2 b^{\mathrm{T}} b > 0$. From (11.10), it is seen that $c^{\mathrm{T}} d - \alpha \leq 0$ for all $x \in B^{-1}(0)$. This implies that $L_{f_{\mathrm{cl}}} B < 0$ for all $x \in B^{-1}(0)$.

By Section 6.5 in [16], α is continuous since D is convex, c is continuous, and α is linear in d for each x. Thus the controller k is continuous, as it is the product of the continuous function ξ and a second function that is continuous at any point except at $b = 0$. Since $\xi(||b||) = 0$ in a neighborhood of $b = 0$, the product is continuous for all x. □

Corollary 15: The properness of B imposed in Proposition 47 can be replaced with compactness of $\{x \in X | b(x) = 0\}$.

To design the control system for the surgical robot, we find one control barrier function per joint of the robot, and use it in the design of a controller. It is possible to completely separate the design of the six controllers, as the reference signal $\theta_{i,\text{ref}}$ for the ith joint angle is generated independent of the other joint angles. Therefore, all control barrier functions are on the following form

$$B_i(x) = (\theta_i - \theta_{i,\text{ref}}(x))^2 - \theta_{i,\text{e}}^2. \tag{11.15}$$

It is seen that the value of B_i is nonpositive when the tracking error is within the specified bound. Since the dynamics of the robot is given as integrators $\dot{\theta}_i = u_i$ for $i = 1, \ldots, 6$, B_i is a control barrier function when $\theta_{i,\text{e}} > 0$. Therefore, we can directly calculate the control laws using Proposition 47 and the designed controllers are safe by design. This finalizes the controller design.

4 Simulation Results

The purpose of this section is to demonstrate how well the designed controllers track the movement of the heart, and keep the tracking error within the specified bounds.

The control barrier functions are chosen to obtain a tracking error less than 0.02 rad of any joint angle. Figure 11.3 shows 5 simulations of the motion compensation with different initial conditions and disturbances to the model. Note that we initialize the system with a nonzero tracking error.

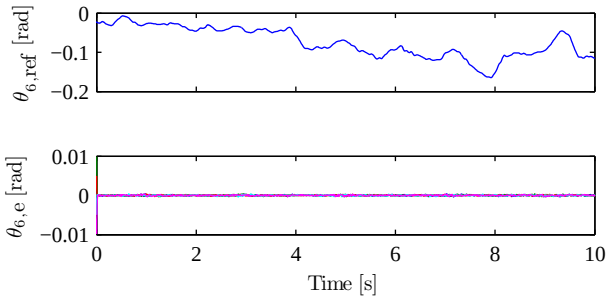


Figure 11.3: The upper subplot shows the reference signal $\theta_{6,\text{ref}}$, and the lower subplot shows the tracking error of θ_6 from different initial conditions.

All trajectories are within the safety bound of 0.02 rad.

The tracking of the position of the tool is shown in Figure 11.4. It demonstrates that the controller puts the surgical tool at a constant relative position with respect to the heart

The tracking error of the position is also within the required bounds. We have omitted a visualization of the tracking of the orientation to save space.

5 Conclusion

In this paper, it was demonstrated that it is possible to derive a safety proof for a robotic surgical system, despite its high complexity. The key to constructing the proof of safety

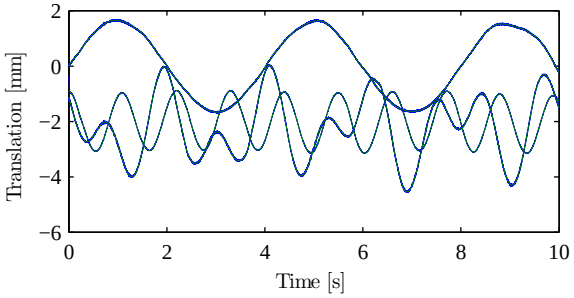


Figure 11.4: Position of the tool (origin of Ψ_7, blue lines) and position of the point of interest on the heart surface (origin of Ψ_h, green lines) in the reference frame Ψ_0.

is to decompose the safety problem into small subproblems that can be solved independently. This allows the generation of multiple control barrier functions, allowing the design of individual controllers for each subsystem.

We have designed and simulated a safe controller for a 6 degree of freedom robot that is capable of compensating for the physiological movements experienced in beating-heart surgery.

The use of safety verification may lead the prevail of autonomous control in robotic surgery, as guaranteed safety will increase the acceptance of sophisticated computer aids in robotic surgery.

A Dynamic Heart Model

In this section, we derive the dynamics of the heart model. First, we provide time-dependent transition matrices; subsequently, we provide the transition matrices as the solution of a system of differential equations. The dynamic heart model is derived from [10, 18].

The movement of the diaphragm is given by a rotation around the x- and y-axis of the reference frame Ψ_0 and a translation along the x- and z-axis of Ψ_0.

The rotation matrix for the diaphragm is

$$R_\mathrm{d}^0 = \begin{bmatrix} \cos\phi_\mathrm{d} & \sin\phi_\mathrm{d} & 0 \\ -\sin\phi_\mathrm{d} & \cos\phi_\mathrm{d} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_\mathrm{d} & \sin\theta_\mathrm{d} \\ 0 & -\sin\theta_\mathrm{d} & \cos\theta_\mathrm{d} \end{bmatrix}, \tag{11.16}$$

where

$$\phi_\mathrm{d}(t) = 3/2\sin(\omega_\mathrm{d} t) \tag{11.17a}$$
$$\theta_\mathrm{d}(t) = -3/2\sin(\omega_\mathrm{d} t) \tag{11.17b}$$
$$\omega_\mathrm{d} = 2\pi/4. \tag{11.17c}$$

The translation of the diaphragm is given by

$$p_\mathrm{d}^0 = \begin{bmatrix} 0 \\ 7/6\cos(2\omega_\mathrm{d} t) - 7/6 \\ 5/3\sin(\omega_\mathrm{d} t) \end{bmatrix}. \tag{11.18}$$

Similarly, the heart rotates around the y- and z-axis of the diaphragm frame Ψ_{d} and translates along its y- and z-axis. This movement is given by

$$R_{\mathrm{h}}^{\mathrm{d}} = \begin{bmatrix} \cos\phi_{\mathrm{h}} & \sin\phi_{\mathrm{h}} & 0 \\ -\sin\phi_{\mathrm{h}} & \cos\phi_{\mathrm{h}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\psi_{\mathrm{h}} & 0 & -\sin\psi_{\mathrm{h}} \\ 0 & 1 & 0 \\ \sin\psi_{\mathrm{h}} & 0 & \cos\psi_{\mathrm{h}} \end{bmatrix} \tag{11.19}$$

and

$$p_{\mathrm{h}}^{\mathrm{d}} = \begin{bmatrix} 15/14\cos(\omega_{\mathrm{h}}t) - 2 \\ -10/9\sin(\omega_{\mathrm{h}}t) - 10/9 \\ 0 \end{bmatrix}, \tag{11.20}$$

where

$$\phi_{\mathrm{h}}(t) = 9/4\cos(\omega_{\mathrm{h}}t) \tag{11.21a}$$
$$\psi_{\mathrm{h}}(t) = 3/8\cos(2\omega_{\mathrm{h}}t) - 9/8 \tag{11.21b}$$
$$\omega_{\mathrm{h}} = 2\pi/1.1. \tag{11.21c}$$

In this work, we are interested in the behavior of the heart expressed in the reference frame Ψ_0. The transformation matrix is calculated from

$${}_{\mathrm{h}}^{0}T(t) = {}_{\mathrm{d}}^{0}T(t)\,{}_{\mathrm{h}}^{\mathrm{d}}T(t), \tag{11.22}$$

and is

$${}_{\mathrm{h}}^{0}T(t) = \begin{bmatrix} {}_{\mathrm{d}}^{0}R(t){}_{\mathrm{h}}^{\mathrm{d}}R(t) & {}_{\mathrm{d}}^{0}p(t) + {}_{\mathrm{d}}^{0}R(t){}_{\mathrm{h}}^{\mathrm{d}}p(t) \\ 0 & 1 \end{bmatrix}. \tag{11.23}$$

The movements described above can be expressed as the solution of a system of polynomial differential equation. It is necessary to find this vector field to accomplish the analysis of this paper. The vector field is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \\ \dot{x}_{13} \\ \dot{x}_{14} \\ \dot{x}_{15} \\ \dot{x}_{16} \end{bmatrix} = \begin{bmatrix} \omega_{\mathrm{d}}x_2 \\ -\omega_{\mathrm{d}}x_1 \\ 2\omega_{\mathrm{d}}x_4 \\ -2\omega_{\mathrm{d}}x_3 \\ 3/2\omega_{\mathrm{d}}x_2x_6 \\ -3/2\omega_{\mathrm{d}}x_2x_5 \\ -3/2\omega_{\mathrm{d}}x_2x_8 \\ 3/2\omega_{\mathrm{d}}x_2x_7 \\ \omega_{\mathrm{h}}x_{10} \\ -\omega_{\mathrm{h}}x_9 \\ 2\omega_{\mathrm{h}}x_{12} \\ -2\omega_{\mathrm{h}}x_{11} \\ -9/4\omega_{\mathrm{h}}x_9x_{14} \\ 9/4\omega_{\mathrm{h}}x_9x_{13} \\ -6/8\omega_{\mathrm{h}}x_{11}x_{16} \\ 6/8\omega_{\mathrm{h}}x_{11}x_{15} \end{bmatrix} + \begin{bmatrix} x_2 & 0 \\ -x_1 & 0 \\ 2x_4 & 0 \\ -2x_3 & 0 \\ 3/2x_2x_6 & 0 \\ -3/2x_2x_5 & 0 \\ -3/2x_2x_8 & 0 \\ 3/2x_2x_7 & 0 \\ 0 & x_{10} \\ 0 & -x_9 \\ 0 & 2x_{12} \\ 0 & -2x_{11} \\ 0 & -9/4x_9x_{14} \\ 0 & 9/4x_9x_{13} \\ 0 & -6/8x_{11}x_{16} \\ 0 & 6/8x_{11}x_{15} \end{bmatrix} d. \tag{11.24}$$

Note that the interpretation of each state is seen from the initial condition provided in (11.25). We show how the rotations and translations can be expressed in terms of the solution of the differential equation.

The rotation matrix R_{d}^{0} and translation p_{d}^{0} are shown in the following in terms of the solution $\phi(t)$ of the vector field from initial condition

$$\phi(0) = \begin{bmatrix} \sin(\omega_{\mathrm{d}}0) \\ \cos(\omega_{\mathrm{d}}0) \\ \sin(2\omega_{\mathrm{d}}0) \\ \cos(2\omega_{\mathrm{d}}0) \\ \sin(3/2\sin(\omega_{\mathrm{d}}0)) \\ \cos(3/2\sin(\omega_{\mathrm{d}}0)) \\ \sin(-3/2\sin(\omega_{\mathrm{d}}0)) \\ \cos(-3/2\sin(\omega_{\mathrm{d}}0)) \\ \sin(\omega_{\mathrm{h}}0) \\ \cos(\omega_{\mathrm{h}}0) \\ \sin(2\omega_{\mathrm{h}}0) \\ \cos(2\omega_{\mathrm{h}}0) \\ \sin(9/4\cos(\omega_{\mathrm{h}}0)) \\ \cos(9/4\cos(\omega_{\mathrm{h}}0)) \\ \sin(3/8\cos(2\omega_{\mathrm{h}}0) - 9/8) \\ \cos(3/8\cos(2\omega_{\mathrm{h}}0) - 9/8) \end{bmatrix} \tag{11.25}$$

$$R_{\mathrm{d}}^{0}(t) = \begin{bmatrix} \phi_6(t) & \phi_5(t) & 0 \\ -\phi_5(t) & \phi_6(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \phi_8(t) & \phi_7(t) \\ 0 & -\phi_7(t) & \phi_8(t) \end{bmatrix}, \tag{11.26}$$

and

$$p_{\mathrm{d}}^{0}(t) = \begin{bmatrix} 0 \\ 7/6\phi_4(t) - 7/6 \\ 5/3\phi_1(t) \end{bmatrix}. \tag{11.27}$$

Similar expressions are set up for the heart

$$R_{\mathrm{h}}^{\mathrm{d}}(t) = \begin{bmatrix} \phi_{14}(t) & \phi_{13}(t) & 0 \\ -\phi_{13}(t) & \phi_{14}(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{16}(t) & 0 & -\phi_{15}(t) \\ 0 & 1 & 0 \\ \phi_{15}(t) & 0 & \phi_{16}(t) \end{bmatrix} \tag{11.28}$$

and

$$p_{\mathrm{h}}^{\mathrm{d}}(t) = \begin{bmatrix} 15/14\phi_{10}(t) - 2 \\ -10/9\phi_9(t) - 10/9 \\ 0 \end{bmatrix}. \tag{11.29}$$

B Kinematic Model of Robot

In this section, the kinematic model of the robot is presented. The considered 6 degree of freedom robot is shown in Figure 11.5.

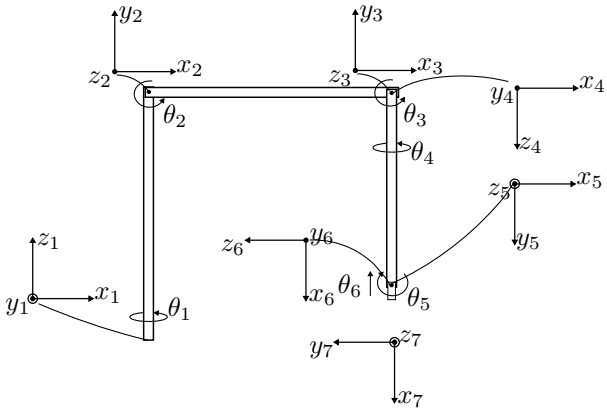


Figure 11.5: Illustration of the considered robot. Coordinate frames and angles are added to the figure, to show how the robot can be controlled.

In the bottom of the robot we put a base frame Ψ_b. We provide all link transformations and a transformation from the wrist frame (Ψ_6) to the base frame. The base frame of the robot Ψ_b is not the reference frame utilized in the design of the controller. This is chosen to be at the initial location of the diaphragm to ease calculations. The link transformations are

$$
{}^{0}_{\mathrm{b}}T = \begin{bmatrix} 1 & 0 & 0 & {}^{0}_{\mathrm{b}}p_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & {}^{0}_{\mathrm{b}}p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30a}
$$

$$
{}^{\mathrm{b}}_{1}T(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30b}
$$

$$
{}^{1}_{2}T(\theta_2) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30c}
$$

$$
{}^{2}_{3}T(\theta_3) = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30d}
$$

$$
{}^{3}_{4}T(\theta_4) = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30e}
$$

$$
{}^{4}_{5}T(\theta_5) = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30f}
$$

$$
{}^{5}_{6}T(\theta_6) = \begin{bmatrix} 0 & 0 & -1 & 0 \\ \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ -\sin\theta_6 & -\cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.30g}
$$

$$
{}^{6}_{7}T = \begin{bmatrix} 1 & 0 & 0 & a_7 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11.30h}
$$

The parameters for the robot are $a_3 = 600$ mm, $a_7 = 14$ mm, $d_2 = 600$ mm, $d_5 = 467$ mm, ${}^{0}_{\mathrm{b}}p_x = 700$ mm, and ${}^{0}_{\mathrm{b}}p_z = 50$ mm.

In the remainder, we use following short hand notation $c_i \equiv \cos\theta_i$, $c_{ij} \equiv \cos(\theta_i + \theta_j)$, $s_i \equiv \sin\theta_i$, and $s_{ij} \equiv \sin(\theta_i + \theta_j)$. A transformation from the wrist frame to the base frame is

$$
\begin{aligned}
&{}^{\mathrm{b}}_{6}T(\theta) = \\
&\begin{bmatrix}
(-(c_1c_{23}c_4 + s_1s_4)s_5 + c_1s_{23}c_5)c_6 - (c_1c_{23}s_4 - s_1c_4)s_6 \\
(-(s_1c_{23}c_4 - c_2s_4)s_5 + s_1s_{23}c_5)c_6 - (s_1c_{23}s_4 - c_2c_4)s_6 \\
(-s_{23}c_4s_5 - c_{23}c_5)c_6 - s_{23}s_4s_6 \\
0
\end{bmatrix} \\
&\begin{matrix}
-(-(c_1c_{23}c_4 + s_1s_4)s_5 + c_1s_{23}c_5)s_6 - (c_1c_{23}s_4 - s_1c_4)c_6 \\
-(-(s_1c_{23}c_4 - c_2s_4)s_5 + s_1s_{23}c_5)s_6 - (s_1c_{23}s_4 - c_2c_4)c_6 \\
(s_{23}c_4s_5 + c_{23}c_5)s_6 - s_{23}s_4c_6 \\
0
\end{matrix} \\
&\begin{bmatrix}
-(c_1c_{23}c_4 + s_1s_4)c_5 - c_1s_{23}s_5 & c_1c_{23}d_5 + c_1c_2a_3 \\
-(s_1c_{23}c_4 - c_2s_4)c_5 - s_1s_{23}s_5 & s_1s_{23}d_5 + s_1c_2a_3 \\
-s_{23}c_4c_5 + c_{23}s_5 & -c_{23}d_5 + s_2a_3 + d_2 \\
0 & 1
\end{bmatrix}.
\end{aligned} \tag{11.31}
$$

C Inverse Kinematics

To control the robot to a desired position and orientation, we derive the six joint angles $\theta_1, \ldots, \theta_6$ from a given reference matrix ${}^{0}_{7}T$. The relation between $\theta_1, \ldots, \theta_6$ and ${}^{0}_{7}T$ is

$$
{}^{0}_{7}T = {}^{0}_{\mathrm{b}}T\,{}^{\mathrm{b}}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6)\,{}^{6}_{7}T, \tag{11.32}
$$

where

$$
{}^{0}_{7}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11.33}
$$

The transformation from the tool frame Ψ_7 to the wrist frame Ψ_6 is shown below

$$
\begin{aligned}
{}^{\mathrm{b}}_{6}T(\theta) &= {}^{0}_{\mathrm{b}}T^{-1}\,{}^{\mathrm{b}}_{7}T^{6}_{7}T^{-1} && \text{(11.34a)}\\
&= \begin{bmatrix} 1 & 0 & 0 & -{}^{0}_{\mathrm{b}}p_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -{}^{0}_{\mathrm{b}}p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \begin{bmatrix} 1 & 0 & 0 & -a_7 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && \text{(11.34b)}\\
&= \begin{bmatrix} r_{11} & -r_{13} & r_{12} & p_x - {}^{0}_{\mathrm{b}}p_x - a_7 r_{11} \\ r_{21} & -r_{23} & r_{22} & p_y - a_7 r_{21} \\ r_{31} & -r_{33} & r_{32} & p_z - {}^{0}_{\mathrm{b}}p_z - a_7 r_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix}. && \text{(11.34c)}
\end{aligned}
$$

For convenience, we introduce the following notation

$$
\begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \bar{r}_{13} & \bar{p}_x \\ \bar{r}_{21} & \bar{r}_{22} & \bar{r}_{23} & \bar{p}_y \\ \bar{r}_{31} & \bar{r}_{32} & \bar{r}_{33} & \bar{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \equiv \begin{bmatrix} r_{11} & -r_{13} & r_{12} & p_x - {}^{0}_{\mathrm{b}}p_x - a_7 r_{11} \\ r_{21} & -r_{23} & r_{22} & p_y - a_7 r_{21} \\ r_{31} & -r_{33} & r_{32} & p_z - {}^{0}_{\mathrm{b}}p_z - a_7 r_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.35a}
$$

$$
{}^{\mathrm{b}}_{6}T \equiv \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \bar{r}_{13} & \bar{p}_x \\ \bar{r}_{21} & \bar{r}_{22} & \bar{r}_{23} & \bar{p}_y \\ \bar{r}_{31} & \bar{r}_{32} & \bar{r}_{33} & \bar{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11.35b}
$$

We find an expression for θ_1, by use of

$$
{}^{\mathrm{b}}_{1}T^{-1}(\theta_1)\,{}^{\mathrm{b}}_{6}T = {}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6), \tag{11.36}
$$

where the right hand side of (11.36) is

$$
\begin{bmatrix} (-c_{23}c_4s_5 + s_{23}c_5)c_6 - c_{23}s_4s_6 \\ s_4s_5c_6 - c_4s_6 \\ (-s_{23}c_4s_5 - c_{23}c_5)c_6 - s_{23}s_4s_6 \\ 0 \end{bmatrix}
\begin{bmatrix} (c_{23}c_4s_5 - s_{23}c_5)s_6 - c_{23}s_4c_6 \\ -s_4s_5s_6 - c_4c_6 \\ (s_{23}c_4s_5 + c_{23}c_5)s_6 - s_{23}s_4c_6 \\ 0 \end{bmatrix}
\begin{bmatrix} -c_{23}c_4c_5 - s_{23}s_5 & c_2a_3 + s_{23}d_5 \\ s_4c_5 & 0 \\ -s_{23}c_4c_5 + c_{23}s_5 & s_2a_3 - c_{23}d_5 \\ 0 & 1 \end{bmatrix}. \tag{11.37}
$$

Therefore, (11.37) is equal to

$$\begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \bar{r}_{13} & \bar{p}_x \\ \bar{r}_{21} & \bar{r}_{22} & \bar{r}_{23} & \bar{p}_y \\ \bar{r}_{31} & \bar{r}_{32} & \bar{r}_{33} & \bar{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11.38}$$

Element (2,4) of the above matrix equality yields

$$\cos(\theta_1)p_y - \sin(\theta_1)p_x = 0. \tag{11.39}$$

Hence, $\theta_1 = \tan^{-1}(p_y/p_x)$.

We derive an expression for θ_3 using elements (1,4) and (3,4) of the above matrix equality

$$c_2 a_3 + s_{23} d_5 = c_1 p_x + s_1 p_y \tag{11.40a}$$
$$s_2 a_3 + d_2 - c_{23} d_5 = p_z. \tag{11.40b}$$

By squaring both equations and summing them, we get an expression for s_3

$$s_3 = \frac{(c_1 p_x + s_1 p_y)^2 + (p_z - d_2)^2 - a_3^2 - d_5^2}{2 a_3 d_5}. \tag{11.41}$$

Hence, $\theta_3 = \sin^{-1}(s_3)$.

To calculate θ_2, we first calculate $\theta_2 + \theta_3$ denoted by θ_{23}. This is calculated from ${}^{\mathrm{b}}_{3}T^{-1}(\theta)$

$${}^{\mathrm{b}}_{3}T^{-1}(\theta) = \begin{bmatrix} c_1 c_{23} & s_1 c_{23} & s_{23} & -d_2 s_{23} - a_3 c_3 \\ -c_1 s_{23} & -s_1 s_{23} & c_{23} & -d_2 c_{23} + a_3 s_3 \\ s_1 & -c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.42}$$

Elements (1,4) and (2,4) result in

$$s_{23} = \frac{a_3 c_3 (p_z - d_2) + (a_3 s_3 + d_5)(c_1 p_x + s_1 p_y)}{(p_z - d_2)^2 + (c_1 p_x + s_1 p_y)^2} \tag{11.43a}$$
$$c_{23} = \frac{a_3 c_3 (c_1 p_x + s_1 p_y) - (a_3 s_3 + d_5)(p_z - d_2)}{(p_z - d_2)^2 + (c_1 p_x + s_1 p_y)^2}, \tag{11.43b}$$

and θ_{23} can be found from

$$\tan(\theta_{23}) = \frac{a_3 c_3 (p_z - d_2) + (a_3 s_3 + d_5)(c_1 p_x + s_1 p_y)}{a_3 c_3 (c_1 p_x + s_1 p_y) - (a_3 s_3 + d_5)(p_z - d_2)}. \tag{11.44}$$

Now, $\theta_2 = \theta_{23} - \theta_3$.

θ_4 is calculated from elements (1,3) and (3,3) of ${}^{\mathrm{b}}_{3}T^{-1}(\theta)$

$$c_1 c_{23} \bar{r}_{13} + s_1 c_{23} \bar{r}_{23} + s_{23} \bar{r}_{33} = -c_4 c_5 \tag{11.45a}$$
$$s_1 \bar{r}_{13} - c_1 \bar{r}_{23} = -s_4 c_5 \tag{11.45b}$$

If $c_5 \neq 0$, then

$$\tan\theta_4 = \frac{-(s_1\bar{r}_{13} - c_1\bar{r}_{23})}{-(c_1c_{23}\bar{r}_{13} + s_1c_{23}\bar{r}_{23} + s_{23}\bar{r}_{33})}. \tag{11.46}$$

We calculate θ_5 from

$${}^{b}_{4}T^{-1}(\theta){}^{0}_{6}T = {}^{4}_{5}T(\theta_5){}^{5}_{6}T(\theta_6), \tag{11.47}$$

where

$${}^{b}_{4}T^{-1}(\theta) = \begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & s_1c_{23}c_4 - c_1s_4 & s_{23}c_4 & -c_1c_2a_3 \\ -c_1c_{23}s_4 + s_1c_4 & -s_1c_{23}s_4 - c_1c_4 & -s_{23}s_4 & -s_1c_2a_3 \\ c_1s_{23}c_4 & s_1s_{23} & -c_{23} & -s_2a_3 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.48}$$

and

$${}^{4}_{5}T(\theta_5){}^{5}_{6}T(\theta_6) = \begin{bmatrix} -s_5c_6 & s_5s_6 & -c_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ c_5c_6 & c_5s_6 & -s_5 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11.49}$$

Elements (1,3) and (3,3) of the above matrix equality are

$$-c_5 = (c_1c_{23}c_4 + s_1s_4)\bar{r}_{13} + (s_1c_{23}c_4 - c_1s_4)\bar{r}_{23} + s_{23}c_4\bar{r}_{33} \tag{11.50a}$$

$$-s_5 = c_1s_{23}\bar{r}_{13} + s_1s_{23}\bar{r}_{23} - c_{23}\bar{r}_{33}. \tag{11.50b}$$

This implies that

$$\tan(\theta_5) = \tan^{-1}\left(\frac{s_5}{c_5}\right). \tag{11.51}$$

We calculate θ_6 from

$${}^{b}_{5}T^{-1}(\theta){}^{b}_{6}T = {}^{5}_{6}T(\theta_6), \tag{11.52}$$

where

$${}^{b}_{5}T^{-1}(\theta) = \begin{bmatrix} (c_1c_{23}c_4 + s_1s_4)c_5 + c_1s_{23}s_5 \\ -(c_1c_{23}c_4 + s_1s_4)s_5 + c_1s_{23}c_5 \\ c_1c_{23}s_4 - s_1c_4 \\ 0 \end{bmatrix} \begin{bmatrix} (s_1c_{23}c_4 - c_1s_4)c_5 + s_1s_{23}s_5 \\ -(s_1c_{23}c_4 - c_1s_4)s_5 + s_1s_{23}c_5 \\ s_1c_{23}s_4 + c_1c_4 \\ 0 \end{bmatrix} \begin{bmatrix} s_{23}c_4c_5 - c_{23}s_5 & -c_1s_{23}d_5 - c_1c_2a_3 \\ -s_{23}c_4s_5 - c_{23}c_5 & -s_1s_{23}d_5 - s_1c_2a_3 \\ s_{23}s_4 & c_{23}d_5 - s_2a_3 - d_2 \\ 0 & 1 \end{bmatrix} \tag{11.53}$$

Elements (2,1) and (3,1) of the above matrix equality are

$$\begin{aligned}&(-(c_1c_{23}c_4+s_1s_4)s_5+c_1s_{23}c_5)\bar{r}_{11}\\&+(-(s_1c_{23}c_4-c_1s_4)s_5+s_1s_{23}c_5)\bar{r}_{21}+(-s_{23}c_4s_5-c_{23}c_5)\bar{r}_{31}=c_6\end{aligned} \tag{11.54a}$$

$$(c_1c_{23}s_4-s_1c_4)\bar{r}_{11}+(s_1c_{23}s_4+c_1c_4)\bar{r}_{21}+s_{23}s_4\bar{r}_{31}=-s_6. \tag{11.54b}$$

This implies that

$$\theta_6=\tan^{-1}\left(\frac{s_6}{c_6}\right). \tag{11.55}$$

References

[1] Y. Nakamura, K. Kishi, and H. Kawakami, "Heartbeat synchronization for robotic cardiac surgery," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2001, pp. 2014–2019.

[2] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis, "Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 7, no. 4, pp. 375–392, 2011.

[3] N. Leveson and C. Turner, "An investigation of the Therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, July 1993.

[4] J. Ding, J. H. Gillula, H. Huang, M. P. Vitus, W. Zhang, and C. J. Tomlin, "Hybrid systems in robotics," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 33–43, September 2011.

[5] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa, "Robotic surgery: Formal verification of plans," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 24–32, September 2011.

[6] F. Kerber and A. van der Schaft, "Compositional analysis for linear control systems," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. New York, NY, USA: ACM, 2010, pp. 21–30.

[7] C. Sloth, G. J. Pappas, and R. Wisniewski, "Compositional safety analysis using barrier certificates," in *Proceedings of Hybrid Systems: Computation and Control*, 2012, pp. 15–23.

[8] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, August 2007.

[9] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," in *Proceedings of the 7th IFAC Symposium on Nonlinear Control Systems*, 2007, pp. 462–467.

[10] G. Shechter, C. Ozturk, J. Resar, and E. McVeigh, "Respiratory motion of the heart from free breathing coronary angiograms," *IEEE Transactions on Medical Imaging*, vol. 23, no. 8, pp. 1046–1056, aug. 2004.

[11] L. W. Sun, F. Van Meer, J. Schmid, Y. Bailly, A. A. Thakre, and C. K. Yeung, "Advanced da Vinci surgical system simulator for surgeon training and operation planning," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 3, no. 3, pp. 245–251, 2007.

[12] E. D. Sontag, "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems & Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.

[13] R. Richa, A. P. L. Bo, and P. Poignet, "Towards robust 3d visual tracking for motion compensation in beating heart surgery," *Medical Image Analysis*, vol. 15, no. 3, pp. 302–315, 2011.

[14] C. Sloth, R. Wisniewski, J. Larsen, J. Leth, and J. Poulsen, "Model of beating-heart and surgical robot," Aalborg University, Tech. Rep., 2012.

[15] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson Prentice Hall, 2005.

[16] J.-P. Aubin, *Viability Theory*. Birkhäuser, 1991.

[17] I. H. Madsen and J. Tornehave, *From Calculus to Cohomology*. Cambridge University Press, 1997.

[18] V. Duindam and S. Sastry, "Geometric motion estimation and control for robotic-assisted beating-heart surgery," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2007, pp. 871–876.