Hybrid Systems: A First-Order Approach to Verification and Approximation Techniques

Candidate: Alberto Casagrande

Supervisor: Prof. Alberto Policriti
Prof. Tiziano Villa

March 31, 2006
Author's e-mail: casagrande@dimi.uniud.it

Author's address:

Dipartimento di Matematica e Informatica
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine
Italia
To my family
Acknowledgments

This thesis is the result of a path which I have walked through since I was a child. For this reason, I would like to thank first of all my family for their support and teachings. My grandmothers Florinda and Norma are living example of great courage: they passed through the majority of last century and they still have the force of going on, teaching me something new every day. My grandfathers Giuseppe and Lorenzo: everyone takes them as examples of integrity and honesty. I miss them so much. My father Carlo and my mother Diana: they have always encouraged and supported me despite my wishes. I could never repay them. And at last, Carla who spurs and motivates me. She is my inspiration and my enthusiasm and I would be a different person without her. I hope she’ll have the patience to bear me also in the future.

I would like to thank my supervisors Prof. Alberto Policriti and Prof. Tiziano Villa for guiding me in the writing of this thesis and during my three years as a PhD student. Much of this work is due to their teachings and it would not be possible without them.

I would like also to express my deepest gratitude to PARADES for hosting and financing me during last three years. I wish to thank Andrea Balluchi, Prof. Alberto Sangiovanni-Vincentelli, and all the PARADES staff for their support: they gave me the unique opportunity of working with some of the most exceptional minds in the world in a really exciting research field. I would like to thank Prof. Bud Mishra for hosting me at the NYU Bioinformatics Group and for giving me a great opportunity of improving my research. Furthermore, I am really indebted with Salvatore “Toto” Paxia for his help during my visit at NYU: without him I would have lived under a bridge. I am also grateful to my numerous coauthors, Andrea Balluchi, Prof. Luca Benvenuti, Pieter Collins, Alberto Ferrari, Prof. Bud Mishra, Venkatesh Mysore, Pierpaolo Murri, Prof. Alberto Policriti, Prof. Carla Piazza, Prof. Alberto Sangiovanni-Vincentelli, and Prof. Tiziano Villa, for the fruitful meetings concerning our work. A special thanks goes to Prof. Agostino Dovier for his comments on this thesis and for his effort in the final publishing phase. And I thank also Prof. Fabio Zanolin and Paolo Baiti for useful discussions about Part II and Oleg Botchkarev and Stavros Tripakis for providing the Verishift source code.

For their friendship, for the wonderful hours spent together, and, more than all, for their ability in enduring me while I am eating potatoes and drinking branca menta, I will never thank enough Andrea also known as “Scarpa”, Christian better known as “Ciccio”, Silvia also known as “Cynzia”, “er (tra poco cattolicissimo)
Max, Valeria, Maura, Flavia, Tommaso, Teresa, Alessio, Manuela, Dino better known as “Pex”, Emanuele also known as “EP”, Paolo also known as “Blues”, Vanni also known as “WAAAAALLLLLLYYYY”, Andrea better known as “Greppo”, Davide Bresolin, Marco better known as “Ottimo”, Simone, Alessandro, Nicola, Pippo, Gabriele also known as “Grego”, Annalisa better known as “Morghi”, all the “Loncata” team, and all my colleagues at the Department of Mathematics and Computer Science.

Finally, I want to relieve all the persons who have born my awful mood during the last months: it has not depended on this thesis, it is my usual disposition.
Abstract

Systems having a mixed discrete-continuous evolution are called hybrid systems. Since hybrid systems cannot be studied by either dynamical system techniques or finite state system approaches only, specific formal tools, hybrid automata, were introduced to model them. Intuitively, a hybrid automaton is a “finite-state” automaton with continuous variables which evolve according to a set of continuous laws. Such kind of automata has been widely used to demonstrate the validity of hybrid system properties and, even if it is proved that many simple verification problems, such as reachability, are not in general decidable over them, various model checking techniques have been proposed in literature. In this dissertation, we further investigate both the notion of hybrid automaton and the model checking problem over such structure. We relate first-order theories and analysis results on multi-valued maps and we reduce bounded reachability problem for hybrid automata whose continuous laws are expressed by inclusions to decidability problem for first-order formulae over the reals. We introduce two classes of hybrid automata for which the reachability problem can be decided and we prove decidability of model checking for a CTLsub-logic over them. We suggest a new algorithm for computing approximations of the reached sets, based on a previous algorithm presented by Botchkarev. We present some theoretical results on termination for both algorithms and we prove that they have the same complexity. Finally, we introduce a new software package, called ARIADNE, for the verification of hybrid automaton properties and we show that, since it relies on a rigorous computable analysis theory to represent geometric objects, it is capable to achieve provable approximation bounds along the computations.
# Contents

<table>
<thead>
<tr>
<th>Table of contents</th>
<th>ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iii</td>
</tr>
<tr>
<td>Introduction</td>
<td>v</td>
</tr>
</tbody>
</table>

## I Basic Concepts

1 Theories and Decidability

1.1 Logic Languages and Theories ............................................ 3
1.1.1 Models and Theories ..................................................... 4
1.2 O-Minimal Theories ........................................................ 5
1.2.1 Semi-Algebraic Theory .................................................. 5
1.2.2 Restricted Analytic Theory ............................................. 6
1.2.3 Exponential Theories ................................................. 7

2 Models and Temporal Logics

2.1 Graphs .............................................................................. 9
2.2 Models and Temporal Logics .............................................. 11
2.3 Discrete Temporal Logics .................................................. 13
2.3.1 Computation Tree Logic* ................................................ 13
2.3.2 LTL and CTL ............................................................... 14
2.4 Dense Temporal Logics ....................................................... 15
2.4.1 CTL* without the $\circ$ Operator ................................. 15
2.4.2 LTL$_{\circ}$ and CTL$_{\circ}$ Logics .................................. 16
2.4.3 Timed CTL Logic ......................................................... 17
2.5 Model Checking ............................................................... 18
2.6 Simulation and Bisimulation ............................................. 19
2.6.1 Simulation ................................................................. 19
2.6.2 Bisimulation ............................................................... 20

3 Hybrid Automata

3.1 Syntax ............................................................................. 23
3.2 Semantics and Reachability .............................................. 25
3.3 Model Checking for Hybrid Systems ................................. 29
3.4 Hybrid Automaton Classes ............................................... 30
3.4.1 Linear Automata ........................................................... 31
3.4.2 O-minimal Automata .................................................... 33
## II Hybrid Automata with Inclusion Dynamics

### 4 Dynamics and Flow Selections

4.1 Dynamics and Selection Problem .................................. 37
4.2 Michael’s Form ...................................................... 39
4.3 Reachability .......................................................... 42
  4.3.1 Reachability for Transitive Dynamics ......................... 46

### 5 First-Order Constant Reset Hybrid Automata

5.1 FOCoRe Definition .................................................. 51
5.2 Reachability and Model Checking .................................. 53
5.3 FOCoRe and Bisimulation ........................................... 56
5.4 Model Checking ...................................................... 63

### 6 Independent Dynamics Hybrid Automata

6.1 Basic Notions ....................................................... 69
6.2 A New Hybrid Automaton Class .................................... 73
6.3 Identifying an IDA .................................................. 76
6.4 Reachability ........................................................ 78
6.5 From Reachability to Satisfiability ............................... 82
6.6 IDA and Bisimulation ............................................... 89

## III Approximating Analysis

### 7 Improving Reachability Analysis Algorithms

7.1 Botchkarev’s Algorithm ............................................. 94
7.2 Some Theoretical Results on Termination ....................... 97
7.3 An Improved Approximation Algorithm ........................ 100
7.4 Test Case Results .................................................... 105

### 8 ARIADNE - A library for hybrid automata

8.1 Computable Analysis and Approximation ....................... 110
  8.1.1 Representation of Points and Sets .......................... 111
  8.1.2 Representation of Functions and Operations ............... 113
8.2 Ariadne’s Implementation ........................................... 114
  8.2.1 The Base Module ............................................. 115
  8.2.2 The Geometry Module ....................................... 116
  8.2.3 The Evaluation Module ...................................... 117
8.3 Comparison with Existing Languages/Tools ..................... 118

Conclusions .............................................................. 123

Bibliography ............................................................ 127

Index ........................................................................... 141
List of Figures

1. A hybrid automaton modelling a thermostat. ............... vi

2.1 A finite model ................................................. 12
2.2 Two equivalent model .................................. 19
2.3 Two simulation equivalent models .................... 22

4.1 A continuous map without continuous selection .......... 39

5.1 The $H_{\infty}$'s dynamics. .............................. 56
5.2 $\text{Pre}_v (G(r))$ ........................................ 62
5.3 $\text{Pre}_v (L(r/3))$ ................................... 63

6.1 Two different touching path components. ............... 70
6.2 The discrete component of an IDA ...................... 76

7.1 A non-Zeno automaton ..................................... 97
7.2 Automaton $\gamma$ ........................................... 101
7.3 Botchkarev’s flow-tube over-approximation. ............ 101
7.4 Engine model .............................................. 106
7.5 Engine controller model .................................. 106
7.6 A partial flow tube representation ..................... 108

8.1 Examples of basic sets. .................................. 112
8.2 Examples of denotable sets. ............................ 113
8.3 An example of approximating set. ....................... 113
8.4 An high-level scheme of ARIADNE. ................... 115
Introduction

“Forty-two!” yelled Loonquawl. “Is that all you’ve got to show for seven and a half million years’ work?”

“I checked it very thoroughly,” said the computer, “and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you’ve never actually known what the question is.”

D. Adams, from The Hitchhiker’s Guide to the Galaxy

During the last century, quantum mechanics described the universe as a discrete environment. However, most of the classical physical models we use to characterise our world are based on the hypothesis that nature evolves in a continuous way. As a matter of fact, the gravity laws, the kinetic theory and the electromagnetic field theory are described using continuous equations over real domains, although each of the phenomena which they model are caused by discrete particles. This approximation simplifies their analysis and allows us to characterise complex events using simple continuous equations. Nevertheless, many systems, such as digital devices, can be described in a very natural way by discrete models, while the corresponding overall continuous models are intractable. Unfortunately, there exist systems having a mixed discrete-continuous behaviour which cannot be characterised in a proper way using either discrete or continuous models. Such systems consist of a discrete program within a continuously changing environment and are named hybrid systems because of their hybrid nature.

Hybrid systems are very common in many fields, such as in automotive, where engine’s physics are guided by a four phase engine, or control theory, where a digital device should be designed to control continuous phenomena. Moreover, hybrid systems can be found in less “canonical” contexts, where the discrete program is not artificial. In particular, they can be found in biological contexts, where the continuous laws change according to a phase cycle. For instance, a cell can be seen as a hybrid system whose substance concentrations depend on the cell’s status.

In order to model hybrid systems, Alur et al. introduced in [4] the notion of hybrid automata. Intuitively a hybrid automaton is a “finite-state” automaton with continuous variables which evolve according to a set of continuous laws, called dynamics, characterising each discrete location. Hybrid automaton states are couple discrete state/continuous variable value and, obviously, they are infinite. A simple example of hybrid automaton representing a thermostat is depicted in Figure 1. In particular,
the modelled thermostat controls a heater and it switches the heater either on, if the temperature is lower than 15°C, or off, if the temperature is higher or equal to 20°C. We present a formal definition of hybrid automaton in Chapter 3.

\[
\begin{align*}
\dot{X} &= -k_r X; & 10 \leq X \leq 30 \\
\dot{X} &= k_h - k_r X; & 10 \leq X \leq 30
\end{align*}
\]

Figure 1: A graphical representation of a hybrid automaton modeling a thermostat.

Traditionally, hybrid automaton dynamics are specified by either differential equations or inclusions: given a differential formula, its solutions are the hybrid automaton’s corresponding dynamics. For instance, if the dynamics in a location is represented by the differential equation \( \dot{x} = \mathcal{G}(x, t) \) and \( f(x, t) \) is solution of such differential equation, then \( x' = f(x, t) \) is the dynamics i.e., \( x' \) can be reached from \( x \) after a \( t \)-timed continuous evolution. An other approach consists in defining the dynamics through a set of formulæ. These formulæ do not involve derivatives and explicitly constraint the hybrid automaton’s evolution.

Specifying the dynamics by differential equations or inclusions has some advantages against a more explicit representations through formulæ. First of all, dynamics usually represent evolution ruled by natural laws and usually physic laws are described by differential equations: specifying dynamics by differential equations does not require any pre-processing in the hybrid automata definition. Moreover, not all differential equations have a computable solution, thus there exist dynamics which can be exactly specified by a differential equation, but not by a formula. Finally, since the solutions of any Cauchy problem are continuous, specifying dynamics by differential equations guarantees the continuity of the dynamics themselves. However, this way of defining dynamics has some drawbacks too. Using formulæ, we can specify dynamics which are not differentiable and we cannot define dynamics whose explicit form is not computable, whereas in the case of dynamics defined by differential equations this can be done. Finally, if we specify dynamics by formulæ, we can exploit quantifier elimination and theories’ decidability to evaluate reachability directly, while using differential equations to define dynamics, we first need some pre-processing to compute the dynamics themselves.

Hybrid system properties can be expressed in a very natural way through modal temporal logics, such as CTL*, CTL (Computation Tree Logic), or LTL (Linear Temporal Logic). Such properties can be verified over hybrid automata using standard finite-state model checking approaches and exploiting equivalence reductions (i.e., simulation or bisimulation) [106, 133, 2] to reduce the number of the automaton’s states to a finite number. In particular, if a hybrid automaton, which has an infinite state space, has either a finite simulation quotient or a finite bisimulation quotient, then
the property can be verified on the reduced model through standard model checking algorithms. Since simulation preserves LTL and bisimulation preserves CTL*, if the property holds on the reduced model, then it also holds on the original hybrid automaton. A central verification problem is the verification of safety properties which requires to check whenever a certain property \( \varphi \) holds during the hybrid automaton’s evolution. Such problem can be naturally reduced to a reachability problem over hybrid automata. To prove that a certain property \( \varphi \) is always true for a hybrid automaton \( H \), we only need to prove that all the states in which \( \varphi \) is false are not reachable by \( H \) from a given set of initial states. Even if it has been proved that reachability problem is not decidable in general, many classes of hybrid automata for which either reachability problem or temporal logic verification is decidable have been proposed. Each of such classes has restrictions on both dynamics and discrete evolutions and they are not suitable to verify properties of many interesting hybrid systems. For these reasons, various works proposed approximations techniques to analyse such systems and many tools, based on such techniques, have been developed in the last years. Unfortunately, all these tools raises two main points. The first issue is that these software packages are usually closed source, and so users can neither customise or optimise them for a specific class of instances of the reachability problem. The second concerns the possibility of obtain a correct arbitrary-precision evaluation of reachable sets.

This dissertation has three main goals. The first one is to introduce hybrid automata, present the most important classes of hybrid automata for which either reachability analysis or model checking problem is decidable, and describe some of the techniques used to decide them.

The second goal is to study hybrid automata whose dynamics are inclusion dynamics defined by formulæ. In particular, we model hybrid automata having dynamics of the type \( x' \in f(x, t) \) and we try to reduce model checking problems over them to decidability problems over first-order formulæ. Such kind of hybrid automata can be used to describe systems of which we have a partial knowledge. For instance, if we want to model the possible interactions of two particles and we do not know their exact trajectories, we may use inclusion dynamics to model all their allowed trajectory. Moreover, inclusion dynamics can be used to estimate unknown parameters of hybrid systems even if their dynamics are not inclusions. For instance, if we want to model a biological hybrid system, we can get system’s behaviours by experiments and we can guess the continuous dynamic type. However, the dynamics themselves may depend of unknown parameters and, to discover them, we can identify all the parameters characterising a hybrid automaton’s behaviour compatible with the experiments. Since we can not test one by one all the possible parameters, we can exploit inclusion dynamics to get the correct ones. Finally, hybrid automata with inclusion dynamics can be used to over-approximate the trajectories of hybrid automata whose dynamics, defined by linear differential equations, cannot be expressed exactly.

Notice that the considered dynamics may not be differentiable, hence, they generalises dynamics defined by differential inclusions. However, we show that imposing continuity to \( f(x, t) \) with respect to \( t \) is not enough to ensure the existence of a proper continuous evolution satisfying the dynamics. For these reasons, we propose a set of
conditions, called Michael’s form, which relate the existence of such evolution to the satisfiability of a first-order formula. Exploiting Michael’s form, we present two classes of hybrid automata for which reachability problems can be reduced to the problem of deciding whenever a logic formula holds or not. Moreover, we show that even if their bisimulation quotient is infinite and the finiteness of their simulation quotient is still an open problem, model checking over a CTL’s sub-logic can be reduced to a decidability problem for first-order formulae too. As a consequence, we get that our decidability results cannot be achieved exploiting standard equivalence reduction techniques such as simulation and bisimulation. Finally, we prove that the problem of deciding whenever a hybrid automaton belongs to such classes can be reduced to the same kind of decidability problem as well. Notice that we focus on computability of the first-order formula reductions and we are not interested on their complexity. In particular, we do not aim at presenting the most efficient reductions, but we just want to show that such reductions can be computed in a effective way.

The last goal this thesis is to both describe a very general approach to approximate reachable region and show how to improve it. Moreover, we present a new tool for reachability problem which exploits computable analysis to provide arbitrary-precision computations of reachable sets.

In particular, this dissertation is split into three parts: in Part I, we describe the basic concepts discussed in the rest of the thesis, we introduce some well known hybrid automaton classes, and we present some interesting techniques to verify properties over them. In Part II, we introduce hybrid automaton classes whose dynamics are inclusion dynamics defined by formulæ and we show how to reduce both reachability and model checking problems over them to first-order formula decidability problems. Finally, in Part III, we discuss about approximating analysis for reachability problem.

More in details, the thesis is organised as follow:

Part I: “Basic Concepts”

Chapter 1 reviews the notion of theory, describes some interesting theories over real numbers, and presents some decidability results over them.

Chapter 2 introduces the basic notations used all over in this dissertation and gives the notion of model. Moreover, it presents temporal logics, model checking, and both simulation and bisimulation notions.

Chapter 3 introduces the formal definition of hybrid automata, presents multirate automata, rectangular hybrid automata, and O-minimal automata, and shows decidability results over them.

Part II: “Hybrid Automata with Inclusion Dynamics”

Chapter 4 shows that not all hybrid automata whose dynamics are continuous have a continuous evolution. Moreover, it proposes a set of conditions, called Michael’s form, which let us reduce the problem of verifying the existence of such evolution to a decidability problem over first-order formulæ and it shows how such conditions can be tested. Finally, it gives an effective reduction from reachability problems over hybrid automata in
Michael’s form to decidability problems for first-order formulæ under the assumption of a finite number of discrete transitions over locations.

Part of the material presented in this chapter has been published in [42, 43].

Chapter 5 introduces a class of hybrid automata, called FOCoRe, which are in Michael’s form. It shows that every FOCoRe’s evolution can be reduced to a FOCoRe’s evolution whose number of discrete transitions is bounded by the number of automaton’s discrete edges and, hence, that reachability problem can be decided. Moreover, it proves that FOCoRe have infinite bisimulation quotient in general, but that model checking over a particular CTL’s sub-logic, called $\Phi_p$, is still decidable.

Part of the material presented in this chapter has been published in [42, 43].

Chapter 6 presents a class of hybrid automata in Michael’s form, called IDA, for which some FOCoRe’s restrictions are relaxed. To maintain the decidability of reachability problem, we require a subset of automaton’s variables to have both identity resets and transitive dynamics. Moreover, for variables in such subset, we require that the corresponding variable dynamics do not change changing location. Finally, this chapter prove that the IDA’s bisimulation quotient can be infinite.

Part of the material presented in this chapter has been published in [41, 40].

Part III: “Approximating Analysis”

Chapter 7 presents a revised reachability computation that avoids the approximations caused by the union operation in the discretised flow-tube estimation.

Part of the material presented in this chapter has been published in [39].

Chapter 8 presents a general open framework for hybrid system verification which has three main goals. First goal is to integrate the existing algorithms and representation techniques into the same frame allowing users to choose the best methods for their needs. An other goal is to provide a development environment, where to implement new space representation techniques and reachability algorithms. The last goal is to implement a tool supporting arbitrary-precision approximate representations of the hybrid automaton evolutions. This framework consists in a library called ARIADNE.

Part of the material presented in this chapter either has been published or is going to be published in [20, 19].

Finally, Chapter contains conclusive considerations about this dissertation and proposes some future works.
I

Basic Concepts
Theories and Decidability

“Quapropter bono christiano sive mathematici sive quilibet inpie divinantium, maxime dicentes vera, cavendi sunt, ne consortio daemoniorum animam deceptam pacto quodam societatis inretiant.”

St. Augustine, from *De Genesi ad Litteram*

In this chapter, we review the notion of first-order theory, we describe some interesting theories and we introduce some decidability results over them. For a more detailed treatment of these notions, the reader should refer to [103, 83, 134].

1.1 Logic Languages and Theories

A first-order language $\mathcal{L}$ is a tuple $\mathcal{L} = \langle \text{Var}, \text{Const}, \text{Funct}, \text{Rel}, \text{PropOp}, \text{Ar} \rangle$, where Var is a set of variables, Const is a set of constant values, Funct is a set of functional operators, Rel is a set of relational symbols, PropOp is a set of propositional operators, and the arity function $\text{Ar} : \text{Funct} \cup \text{Rel} \cup \text{PropOp} \mapsto (\mathbb{N} \setminus \{0\})$ associates to each element of Funct, Rel, and PropOp the number of arguments it takes.

A term of $\mathcal{L}$ can be defined as:

$$\text{term ::= } X | c | \bigwedge_{\text{term}_1, \ldots, \text{term}_{\text{Ar}(\neg)}}$$

where $X$ is a variable in Var, $c$ is a constant in Const, and $\bigwedge$ is a function in Funct.

An atomic formula $\varphi_a$ of $\mathcal{L}$ has the form $\text{tt}$ or $\text{ff}$ or $\varphi_i (\text{term}_1, \ldots, \text{term}_{\text{Ar}(\varphi_i)})$, where $\varphi_i$ is a relational operator in Rel and term is a term of $\mathcal{L}$ for all $i \in [1, \text{Ar}(\varphi_i)]$. Moreover, a formula $\varphi$ of $\mathcal{L}$ is defined as follows:

$$\varphi ::= \varphi_a | \oplus (\varphi_1, \ldots, \varphi_{\text{Ar}(\oplus)}) | \forall X \varphi | \exists X \varphi$$

where $\varphi_a$ is an atomic formula of $\mathcal{L}$, $\oplus$ is a propositional operator in PropOp, $X$ is a variable in Var, and $\varphi_i$ is a formula of $\mathcal{L}$ for all $i \in [1, \text{Ar}(\oplus)]$. The two symbols $\exists$ and $\forall$ are called quantifiers. Usually Rel $= \{\geq\}$ and PropOp $= \{\neg, \lor\}$. 

An occurrence of a variable $X \in \text{Var}$ is \textit{bounded} or \textit{quantified} in a formula $\varphi$, if it occurs in a $\varphi$'s sub-formula of the kind either $\forall X \varphi$ or $\exists X \varphi$. If all the occurrences of a variable $X$ in a formula $\varphi$ are not quantified, then $X$ is said free in $\varphi$. A \textit{sentence} is a formula such that all the variable occurrences are bounded. The set free variables of the first-order formula $\varphi$ is denoted by $\text{Free}(\varphi)$. We will use the notation $\varphi[X_1, \ldots, X_n]$ to stress the fact that $\text{Free}(\varphi)$ is included in the set of variables $\{X_1, \ldots, X_n\}$. By extension, if $\{X_1, \ldots, X_n\}$ is a set of variable vectors, $\varphi[X_1, \ldots, X_n]$ indicates that the free variables of $\varphi$ are included in the set of components of $X_1, \ldots, X_n$. Furthermore, given a formula $\varphi[X_1, \ldots, X_1, \ldots, X_n]$ and a vector $p$ of the same dimension as the variable vector $\bar{X}$, the formula obtained by component-wise substitution of $X_i$ with $p$ is denoted by $\varphi[\bar{X}, \ldots, \bar{X}_{i-1}, p, \bar{X}_{i+1}, \ldots, \bar{X}]$.

### 1.1.1 Models and Theories

A \textit{model} of a language $\mathcal{L}$ is tuple $(M, \text{Const}, \text{Funct}, \text{Rel}, \text{PropOp})$ where:

- $M$ is a nonempty set said support;
- $\text{Const} : \text{Const} \mapsto C \subseteq M$ is an interpretation for $\text{Const}$;
- $\text{Funct} : \text{Funct} \mapsto \bigcup_{k=1}^{\infty} \left( \prod_{i=1}^{k} M \mapsto M \right)$, with $\text{Funct}(\forall) : \prod_{i=1}^{\infty} M \mapsto M$, is an interpretation for $\text{Funct}$;
- $\text{Rel} : \text{Rel} \mapsto \bigcup_{k=1}^{\infty} \left( \prod_{i=1}^{k} M \mapsto \{\text{tt}, \text{ff}\} \right)$, with $\text{Rel}(\equiv) : \prod_{i=1}^{\infty} M \mapsto \{\text{tt}, \text{ff}\}$, is an interpretation for $\text{Rel}$;
- $\text{PropOp} : \text{PropOp} \mapsto \bigcup_{k=1}^{\infty} \left( \prod_{i=1}^{k} \{\text{tt}, \text{ff}\} \mapsto \{\text{tt}, \text{ff}\} \right)$ is an interpretation for $\text{PropOp}$, where $\text{PropOp}(P) : \prod_{i=1}^{\infty} \{\text{tt}, \text{ff}\} \mapsto \{\text{tt}, \text{ff}\}$ for each propositional operator $P$.

The semantics of $\mathcal{L}$ follows from its model directly.

Given a language $\mathcal{L}$ and a model $\mathcal{M}$ of $\mathcal{L}$, we say that a formula $\varphi[X_1, \ldots, X_n]$ in $\mathcal{L}$ is \textit{satisfiable} in $\mathcal{M}$ if there exist $q_1, \ldots, q_n \in M$ such that $\varphi[q_1, \ldots, q_n]$ holds in $\mathcal{M}$. Moreover, we say that $\varphi[X_1, \ldots, X_n]$ is \textit{valid} if $\varphi[q_1, \ldots, q_n]$ holds in $\mathcal{M}$ for all $q_1, \ldots, q_n \in M$. In the following chapters we use the notation $\varphi_1 \equiv \varphi_2$, where $\varphi_1$ and $\varphi_2$ are first-order formulas, meaning that, for all $q_1, \ldots, q_n \in M$, the formula $\varphi_1[q_1, \ldots, q_n]$ holds in $\mathcal{M}$ if and only if $\varphi_2[q_1, \ldots, q_n]$ does too.

If $\text{PropOp}$ and $\text{PropOp}_d$ are not specified, we assume that $\text{PropOp} = \{\lor, \neg\}$ with the usual semantics and, furthermore, talking of models over $S$, where $S$ is a nonempty set, we refer to that models whose $M$ is $S$. Moreover, when $\text{Const} : \text{Const} \mapsto C$ is obvious by the context, we use $(M, C, \text{Funct}, \text{Rel})$ meaning $(M, \text{Const}, \text{Funct}, \text{Rel})$.

\textbf{Example 1.1.1} Consider the language $\mathcal{L}_R \overset{\text{def}}{=} (\text{Var}, \mathbb{Z}, \{+,*\}, \{\geq\}, \text{Ar})$. A model for the language $\mathcal{L}_R$ is the tuple $(\mathbb{R}, \mathbb{Z}, \text{Funct}, \text{Rel})$ where $\text{Funct}$ and $\text{Rel}$ are the usual interpretations for $\{+,*\}$ and $\{\geq\}$, respectively.
1.2. O-Minimal Theories

Notice that such model is equivalent to $\mathcal{M}_0 \overset{\text{def}}{=} (\mathbb{R}, \{0, 1\}, \text{Funct}, \text{Rel})$ in the sense that for each formula $\varphi_R$ in the language $\mathcal{L}_R$ there exists a formula $\varphi_0$ in the language $\mathcal{L}_0 \overset{\text{def}}{=} \langle \text{Var}, \{0, 1\}, \{+, \ast\}, \{\geq\}, \text{Ar} \rangle$ such that $\varphi_R$ holds in $\mathcal{M}_R \overset{\text{def}}{=} (\mathbb{R}, \mathbb{Z}, \text{Funct}, \text{Rel})$ if and only if $\varphi_0$ holds in $\mathcal{M}_0 \overset{\text{def}}{=} (\mathbb{R}, \{0, 1\}, \text{Funct}, \text{Rel})$.

A theory is a set of sentences. Given a language $\mathcal{L}$ and a model $\mathcal{M}$ for it, we can define the theory of of all the sentences of $\mathcal{L}$ which hold in $\mathcal{M}$. Given a model $(\mathcal{M}, C, \text{Funct}, \text{Rel})$, we indicate the corresponding theory by either $(\mathcal{M}, C, \text{Funct}, \text{Rel})$ or $(\mathcal{M}, C, f_0, \ldots, f_n, r_0, \ldots, r_m)$, where $\text{Funct} = \{f_0, \ldots, f_n\}$ and $\text{Rel} = \{r_0, \ldots, r_m\}$.

A set $Y \subseteq M^n$ is said to be definable in a language $\mathcal{L}$ if there exists a formula $\varphi$ in $\mathcal{L}$ with free variables $X_1, \ldots, X_n$ such that $Y = \{\langle q_1, \ldots, q_n \rangle \in M^n \mid \varphi[q_1, \ldots, q_n] \text{ holds}\}$. Notice that every theory fixes the definable sets of a particular language, hence the definability of every set depends on the considered theory.

A theory $\mathcal{T}$ is said model complete (see [124]), if, for every formula $\varphi[X_1, \ldots, X_n]$, there exists an existential formula $\varphi_\exists[X_1, \ldots, X_n]$ such that $\varphi_\exists$ holds in $\mathcal{T}$ if and only if $\varphi$ does too. Moreover, we say that $\mathcal{T}$ admits the elimination of quantifiers, if, for any formula $\varphi[X_1, \ldots, X_n]$, there exists a quantifier free formula $\varphi_0[X_1, \ldots, X_n]$ such that $\varphi$ holds in $\mathcal{T}$ if and only if $\varphi_0$ does. If there exists an algorithm for deciding whether a formula $\varphi$ is satisfiable or not in $\mathcal{T}$, we say that $\mathcal{T}$ is decidable.

Example 1.1.2 Consider the formula $\varphi \overset{\text{def}}{=} \exists X aX^2 + bX + C = 0$. It is well known that, in the theory of reals with $+, \ast, \text{ and } \geq$, $\varphi$ holds if and only if the formula $b - 4ac \geq 0$ holds.

1.2 O-Minimal Theories

An interesting class of theories is the class of O-minimal theories [139, 136].

Definition 1.2.1 (O-Minimal Theory) A theory $\mathcal{T}$ is order minimal, or simply O-minimal, if every set definable in $\mathcal{T}$ is a finite union of points and intervals.

The class of O-minimal theories includes many interesting theories over $\mathbb{R}$. In following subsections, we present some of them.

1.2.1 Semi-Algebraic Theory

The theory $\mathbb{R} = (\mathbb{R}, 0, 1, +, \ast, \geq)$ is called semi-algebraic theory. In [130], Tarski showed that such theory admits the elimination of quantifiers and that it is decidable. Unfortunately, the Tarski’s algorithm has a computational complexity which could not even be expressed as a bounded tower of exponents of the input size. In [50], Collins presented an algorithm, called Cylindrical Algebraic Decomposition (CAD) to decide the satisfiability of a formula $\varphi$ of $\mathcal{L}_R$. Later Hoon Hong, using many useful and practical heuristics, created the first practical quantifier elimination software Qepcad. Alternative CAD-based methods have been proposed Grigorév [70, 71] and Renegar [121, 122, 123] that are doubly exponential in the number of quantifier alternations.
rather than the number of variables. New quantifier elimination approaches have been proposed by Basu, Pollack, and Roy in \[21, 22, 23\]. The total time complexity (bit-complexity) \[109\] of the mentioned semi-algebraic decision procedures are reported in Table 1.1, in the hypothesis that the coefficients of the polynomials can be stored with at most \(B\) bits and that the input formulæ have the form:

\[(Q_1 X[\text{l}]) (Q_2 X[\text{d}]) \ldots (Q_\text{l} X[\text{l}]) (\varphi[X[\text{l}], \ldots, X[\text{l}]])\]

where \(Q_i \in \{\forall, \exists\}\) and \(Q_i \neq Q_{i+1}\), \(X[\text{i}]\) is a partition of all the variables in \(\varphi\), with \(|X[\text{i}]| = n_i\), and \(\varphi\) is a quantifier-free formula with atomic formulæ consisting in \(m\) polynomials of equalities and inequalities of total degree \(d\) having the form

\[g_k(X[\text{1}], \ldots, X[\text{l}]) \geq 0, \quad k = 1, \ldots, m.\]

<table>
<thead>
<tr>
<th>General or Existential</th>
<th>Time Complexity</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>(B^3 (md)^{d(n_i)})</td>
<td>[50]</td>
</tr>
<tr>
<td>Existential</td>
<td>(B^{O(1)} \cdot (md)^{O(n^2)})</td>
<td>[71]</td>
</tr>
<tr>
<td>General</td>
<td>(B^{O(1)} \cdot (md)^{(\sum n_i)^i} \cdot 2)</td>
<td>[70]</td>
</tr>
<tr>
<td>Existential</td>
<td>(B^{1+o(1)} \cdot (md)^{(1+\sum n_i)} d^{O(\sum n_i)^2})</td>
<td>[37, 38]</td>
</tr>
<tr>
<td>General</td>
<td>((B \log B \log \log B)(md)^{(2^{O(1)} \cdot \prod n_i)})</td>
<td>[121, 122, 123]</td>
</tr>
<tr>
<td>Existential</td>
<td>((B \log B \log \log B)(m/n_i)^{d^{O(\sum n_i)}\prod n_i})</td>
<td>[22, 23]</td>
</tr>
<tr>
<td>General</td>
<td>((B \log B \log \log B)(m^\Pi n_i + 1) d^{O(n_i)})</td>
<td>[22, 23]</td>
</tr>
</tbody>
</table>

Table 1.1: Decision procedure complexity for \((\mathbb{R}, 0, 1, +, *, \geq)\).

### 1.2.2 Restricted Analytic Theory

Let \(\mathbb{R}_{an}\) the set of all the real-analytic function from \([-1,1]^n\) to \(\mathbb{R}\). Notice that if \(f : \mathbb{R}^n \rightarrow \mathbb{R}\) is real-analytic in a neighbourhood of \([-1,1]^n\), then the function \(\hat{f}\) defined as

\[
\hat{f}(x) = \begin{cases} 
  f(x) & \text{if } x \in [-1,1]^n \\
  0 & \text{otherwise}
\end{cases}
\]

is real-analytic as well. Consider the theory \(\mathbb{R}_{an} = (\mathbb{R}, 0, 1, +, *, (f)_{f \in \mathbb{R}_{an}}, \geq)\) obtained from \((\mathbb{R}, 0, 1, +, *, \geq)\) adding all the functions in \(\mathbb{R}_{an}\). This theory can describe the behaviour of some periodic trajectories such as sine and cosine functions in a bounded interval. Van den Dries noticed in [135] and that \(\mathbb{R}_{an}\) is model complete. Hence, by Khovanskii’s finiteness theorem (see [86]), \(\mathbb{R}_{an}\) is also O-minimal. Moreover, Denef and Van den Dries gave in [57] a proof of model completeness and O-minimality of \(\mathbb{R}_{an}\) using Weirstrass preparation theorem. Finally, in [137] it was shown that this theory admit the elimination of quantifiers after adding the function \(1/x\) (with \(1/0 = 0\)).
1.2. O-Minimal Theories

1.2.3 Exponential Theories

Another interesting theory is $\mathbb{R}_{\text{exp}} = (\mathbb{R}, 0, 1, +, *, e^x, \geq)$ which is obtained by $(\mathbb{R}, 0, 1, +, *, \geq)$ adding the exponential function $e^x$. Wilkie showed in [141] that this theory is model complete and, thanks to Khovanskii’s results [86], that it is O-minimal. Moreover, in [136] van den Dries proved that an extension of $(\mathbb{R}, 0, 1, +, *, \geq)$ by a family of total real analytic functions admits the elimination of quantifiers if and only if such functions are semi-algebraic. Furthermore, Macintyre and Wilkie presented in [101] an algorithm to decide $\mathbb{R}_{\text{exp}}$ provided that Schanuel’s conjecture (see [44, 100]) holds.

In [138], Wilkie’s method and Khovanskii’s results are used to prove that the semi-algebraic theory extended by exponential operator and analytic functions, $\mathbb{R}_{\text{an,exp}} = (\mathbb{R}, 0, 1, +, *, (f)_{f \in \text{an}}, e^x, \geq)$, is model complete and O-minimal. In [137], a different proof of these properties is given and it is proved also that the theory $\mathbb{R}_{\text{an,exp,log}} = (\mathbb{R}, 0, 1, +, *, (f)_{f \in \text{an}}, e^x, \log x, \geq)$ admits the elimination of quantifiers. Recently, Lion and Rolin gave a geometric proof of $\mathbb{R}_{\text{an,exp}}$’s O-minimality and model completeness in [98].

Finally, in [142], Wilkie gave sufficient and necessary conditions for an extension of semi-algebraic theory by total $C^\infty$ functions to be O-minimal. In particular, semi-algebraic theory extended by total $C^\infty$ Pfaffian functions is O-minimal.
1. Theories and Decidability
Models and Temporal Logics

"Thus mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true."

B. Russell

In this chapter we introduce the basic notations used all over this dissertation and we present some notions, such as *model* or *temporal logics*, which are used to model systems.

2.1 Graphs

Directed graphs are used as discrete components in some of the following notions, hence we start introducing some basics about directed graphs.

**Definition 2.1.1 (Directed Graph)** A directed graph is a pair \( \langle V, E \rangle \) where \( V \) is a finite set of vertices and \( E \subseteq V \times V \) is a set of edges.

If \( \langle v, v' \rangle \in E \) is an edge, then we say that \( v \) and \( v' \) are the source and the destination of \( \langle v, v' \rangle \), respectively.

In this thesis, when we refer to graphs we always intend directed graphs.

In the case of undirected graphs a graph is connected if its vertices are mutually reachable. A directed graph is connected if the undirected graph which can be obtained by considering the edges as unordered pairs is connected. The following definition allows us to formalise the notion of connected graph without passing to its undirected version.

**Definition 2.1.2 (Connected)** Let \( G = \langle V, E \rangle \) be a graph. \( G \) is said to be connected if and only if for each partition \( P = \{ P_1, P_2 \} \) of \( V \), there exist \( v_1 \in P_1 \) and \( v_2 \in P_2 \) such that either \( \langle v_1, v_2 \rangle \in E \) or \( \langle v_2, v_1 \rangle \in E \).

A subgraph of a given graph is obtained by considering a subset of nodes together with the induced subset of edges.
Definition 2.1.3 (Subgraph) Let \( G = (\mathcal{V}, \mathcal{E}) \) be a graph. A graph \( G' = (\mathcal{V}', \mathcal{E}') \) is a subgraph of \( G \) if and only if both \( \mathcal{V}' \subseteq \mathcal{V} \) and \( \mathcal{E}' = \mathcal{E} \cap (\mathcal{V}' \times \mathcal{V}') \) hold.

We are interested in pairs of edges which can be traversed one after the other. These are characterised by the following definition.

Definition 2.1.4 (Subsequent Edges) Let \( G = (\mathcal{V}, \mathcal{E}) \) be a graph. The edge \( e' \) is subsequent to the edge \( e \) if and only if the destination of \( e \) is the source of \( e' \). If \( e' \) is subsequent to \( e \), then the destination of \( e \) is said to be the bridge vertex from \( e \) to \( e' \).

Two edges \( e \) and \( e' \) are said to be subsequent if either \( e \) is subsequent to \( e' \) or \( e' \) is subsequent to \( e \).

From now on we use the notation \( Sb(G) \) to denote the set of edge pairs, \( \langle e, e' \rangle \), of a graph \( G \) such that \( e' \) is subsequent to \( e \).

A path is nothing but a sequence of nodes connected by edges.

Definition 2.1.5 (Path) Let \( G = (\mathcal{V}, \mathcal{E}) \) be a graph. An infinite path of \( G \) is an infinite sequence \( (v_i)_{i \in \mathbb{N}} \) such that, for all \( i \in \mathbb{N} \), \( (v_i, v_{i+1}) \) is an arc of \( G \) i.e., \( (v_i, v_{i+1}) \in \mathcal{E} \).

A finite sequence \( (v_i)_{i \in [0,m]} \) is a path of length \( n \) from \( v_0 \) to \( v_n \) in \( G \) if and only if for all \( i \in [0, n-1] \), it holds \( (v_i, v_{i+1}) \in \mathcal{E} \).

In the following chapters, we denote a finite path \( (v_i)_{i \in [0,m]} \) using also the notation \( (v_0, \ldots, v_n) \). Moreover, we use the notation \( |p| \) to denote the length of the path \( p \) and \( P_\mathcal{E} \) to indicate the set of all finite paths in the graph \( (\mathcal{V}, \mathcal{E}) \). Furthermore, we denote the set of \( (\mathcal{V}, \mathcal{E}) \)'s paths having length at most \( m \) using the notation \( P_\mathcal{E}^{[m]} \). And we indicate the set of \( P_\mathcal{E}^{[m]} \)'s paths starting in \( v \) by \( P_\mathcal{E}^{[m]}(v) \). Now we can define path concatenation.

Definition 2.1.6 (Path Concatenation) Let \( G = (\mathcal{V}, \mathcal{E}) \) be a directed graph. The path concatenation is a function \( \cdot : P_\mathcal{E} \times P_\mathcal{E} \mapsto P_\mathcal{E} \) such that, if \( p = \langle v_0, \ldots, v_n \rangle \) and \( p' = \langle v'_0, \ldots, v'_m \rangle \):

\[
p \cdot p' = \begin{cases} (v_0, \ldots, v_n, v'_1, \ldots, v'_m) & \text{if } v_n = v'_0 \\ p & \text{otherwise.} \end{cases}
\]

In the remaining part of the thesis, we use the notation \( p_1 \cdot p_2 \cdot \ldots \cdot p_m \) meaning \( (((p_1 \cdot p_2) \cdot \ldots) \cdot p_m) \). Moreover, if \( p = \langle v_0, \ldots, v_m \rangle \), then we write \( p^n \) meaning \( p \cdot p^{n-1} \), if \( n > 0 \), or \( \langle v_0 \rangle \) otherwise.

A node \( v \) reaches a node \( v' \) if there is a path from \( v \) to \( v' \). We call this kind of reachability graph reachability to distinguish it from the notion of reachability on hybrid automata which we will introduce in Chapter 3.

Definition 2.1.7 (Graph Reachability) Let \( G = (\mathcal{V}, \mathcal{E}) \) be graph and \( v, v' \in \mathcal{V} \). If there exists a path from \( v \) to \( v' \) in \( G \), then \( v' \) is reachable from \( v \) in \( G \).
2.2 Models and Temporal Logics

To verify the behaviour of a system we need a formal model of it and a way to indicate its properties. In this section we introduce the concepts of model and temporal logic which are used to formally specify a system and to denote the system’s properties, respectively.

We will use $S$ to indicate a set of propositional symbols and, for any set $X$, $2^X$ to indicate the powerset of $X$.

**Definition 2.2.1 (Model)** A model, $M$, is a tuple $M = \langle Q, \Delta_C, \Delta_D, \rightarrow, \Lambda_S \rangle$ where:

- $Q$ is a non empty set of states, either finite or infinite;
- $\Delta_C$ and $\Delta_D$ are the sets of continuous labels and of discrete labels, respectively.
  The set $\Delta \overset{\text{def}}{=} \Delta_C \cup \Delta_D$ is said edge's label set;
- $\rightarrow$ is a transition relation on $Q$ and $\Delta$. In particular, $\rightarrow \subseteq Q \times \Delta \times Q$. A tuple $e \in \rightarrow$ is called edge or arc of the model. We will write $q \xrightarrow{\alpha} q'$ meaning $\langle q, \alpha, q' \rangle \in \rightarrow$;
- $\Lambda_S \colon Q \rightarrow 2^S$ is a labeling function which labels each state with the set of propositional symbols true in that state.

A finite model is a model such that its set of states is finite, whereas an infinite model has an infinite set of states.

Notice that models are usually called Kripke structures in the literature (see [34]). However, there is some confusion on the concept of Kripke structure since it is defined as a finite model in [49, 47, 120], while an infinite set of states is allowed in [34]. To avoid this ambiguity, we always use the term “model”, according to Definition 2.2.1, in place of “Kripke structure”.

When $\Delta$ is a singleton, we omit it from both model’s and transition’s definitions and from graphical representations. In particular, if $\Delta$ is the singleton $\{\alpha\}$, we denote the model $M = \langle Q, \Delta_C, \Delta_D, \rightarrow, \Lambda_S \rangle$ as $M' = \langle Q, \rightarrow', \Lambda_S \rangle$ where $q \xrightarrow{\alpha} q'$ if and only if $q \xrightarrow{\alpha} q'$. Moreover, without loss of generality, we assume that model’s transition relations are total i.e., for every $q \in Q$ there exists a $q' \in Q$ such that $q \rightarrow q'$. Notice that some of the notions introduced in Section 2.1 can be extended to models in a very natural way. In particular, given a model $M = \langle Q, \Delta_C, \Delta_D, \rightarrow, \Lambda_S \rangle$, a path of $M$ is an infinite path of the graph $G_M = \langle Q, \leftrightarrow \rangle$, where $\leftrightarrow = \{(q, q') \mid \exists \sigma \in \Delta \langle q, \sigma, q' \rangle \in \rightarrow\}$. Furthermore we can define the relation $\rightarrow^n \subseteq Q \times Q$, where $n \in \mathbb{N}$ as

\[
\langle q, q' \rangle \in \rightarrow^n \text{ if and only if } \begin{cases} (n = 0 \land q = q') \\ (n > 0 \land \exists q'' \in Q (\langle q, q'' \rangle \in \rightarrow^{n-1} \land \langle q'', q' \rangle \in \rightarrow)) \end{cases}
\]

and the transitive closure $\rightarrow^* \subseteq Q \times Q$ of $\rightarrow$ as the relation $\rightarrow^* = \{(q, q') \mid \exists n \in \mathbb{N} \langle q, q' \rangle \in \rightarrow^n\}$. 


Example 2.2.2 The tuple $\mathcal{M}_0 = \langle Q, \rightarrow, \Lambda_S \rangle$, where:

- $S$ is the set $\{A, B\}$
- $Q$ is the set $\{q_0, q_1, q_2, q_3\}$
- $\rightarrow$ is the relation $\{(q_0, q_1), (q_1, q_2), (q_2, q_0), (q_2, q_1), (q_3, q_3)\}$
- $\Lambda_S$ is such that $\Lambda_S(q_0) = \emptyset$, $\Lambda_S(q_1) = \{A\}$, $\Lambda_S(q_2) = \{A, B\}$, and $\Lambda_S(q_3) = \{B\}$

is a finite model.

Figure 2.1: A graphical representation of the finite model described in Example 2.2.2

![Graphical representation of the finite model](image)

Definition 2.2.3 (Timed Model) A model $\mathcal{M}_0 = \langle Q, R_{\geq 0}, \Delta_D, \rightarrow, \Lambda_S \rangle$ such that, for all $q, q' \in Q$, all $t \in R_{> 0}$, and all $t' \in (0, t)$:

- $q \xrightarrow{0} q'$
- if $q \xrightarrow{t} q'$, then there exist a $q'' \in Q$ such that $q \xrightarrow{t'} q''$

is said to be a timed model.

To specify model’s properties we use a class of logics called temporal logics [116, 117, 118]. Such formalisms extend the propositional logic and express properties of transition sequences. In particular, besides using atomic propositions and traditional Boolean connectives, temporal logics may specify time properties like either “eventually property $p$ will hold”, “from now on property $q$ holds” or “property $r$ will never hold”. These time properties are described using specific temporal operators and path quantifiers which depend on the particular temporal logic adopted.

In this dissertation we use two path quantifiers: $\forall$ (“for all paths”) and $\exists$ (“for some paths”). Intuitively, the former quantifier specifies that all of the computation paths starting at the referred state have a particular property, while the latter denotes that some of that computations have that property.

Moreover, we use five temporal operators: $\circ$ (next), which denotes that a property holds in the next state of the path, $\mathcal{U}$ (until), which holds if the first property holds until the second does, $\Diamond$ (eventually), which specifies a property which holds at some
2.3. Discrete Temporal Logics

2.3.1 Computation Tree Logic*

CTL* distinguishes between two types of properties: properties which are true in a specific state and properties which hold along a specific path. This property classification is introduced in the syntax. In particular, there are two kinds of formulæ in CTL*: state formulæ and path formulæ.

Definition 2.3.1 (CTL* - Syntax) Let $S$ be a set of propositional symbols and $P \in S$. CTL*’s state formulæ are syntactically derived as:

$$\varphi ::= \text{tt} \mid P \mid \neg \varphi \mid \varphi \lor \varphi \mid A \pi$$

where $\pi$ is a CTL*’s path formula, defined by:

$$\pi ::= \varphi \mid \neg \pi \mid \pi \lor \pi \mid o \pi \mid \pi U \pi$$

To ease formula writing, we add some shorthands to syntax. In particular, we use $\text{ff}$, $\varphi_1 \land \varphi_2$, $\varphi_1 \lor \varphi_2$, $\varphi_1 \Rightarrow \varphi_2$, $\varphi_1 \land \varphi_2$, $\diamond \varphi$, $\Box \varphi$, and $\forall \varphi$ to abbreviate $\neg \text{tt}$, $\neg (\neg \varphi_1 \lor \neg \varphi_2)$, $\neg \varphi_1 \lor \varphi_2$, $(\varphi_1 \Rightarrow \varphi_2) \land (\varphi_2 \Rightarrow \varphi_1)$, $\neg (\neg \varphi_1 \lor \neg \varphi_2)$, $\text{tt} U \varphi$, $\neg (\text{tt} U \neg \varphi)$, and $\neg (A \neg \varphi)$, respectively.

Definition 2.3.2 (CTL* - Semantics) Let $M$ be a model and $q$ be a state of $M$. We say that $q$ satisfies the CTL* state formula $\varphi$ on $M$, denoted by $M, q \models \varphi$, if:

- $M, q \models \text{tt}$ always holds;
- $M, q \models P$ if and only if $P \in \Lambda_s(q)$;
- $M, q \models \neg \varphi$ if and only if $M, q \models \varphi$ does not hold;
- $M, q \models \varphi_1 \lor \varphi_2$ if and only if $M, q \models \varphi_1$ or $M, q \models \varphi_2$;
- $M, q \models A \pi$ if and only if $M, (q_i)_{i \in \mathbb{N}} \models \pi$ for all path of the form $(q_i)_{i \in \mathbb{N}}$ such that $q_0 = q$.

Moreover, if $(q_i)_{i \in \mathbb{N}}$ is a path in $M$, we say that $(q_i)_{i \in \mathbb{N}}$ satisfies the CTL* path formula $\pi$ on $M$, denoted by $M, (q_i)_{i \in \mathbb{N}} \models \pi$ as follows:

- $M, (q_i)_{i \in \mathbb{N}} \models \varphi$ if and only if $M, q_0 \models \varphi$;
• $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \neg \pi$ if and only if $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \pi$ does not hold;

• $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \pi_1 \lor \pi_2$ if and only if $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \pi_1$ or $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \pi_2$;

• $\mathcal{M}, (q_i)_{i \in \mathbb{N}} \models \circ \pi$ if and only if $\mathcal{M}, (q_i)_{i \in (\mathbb{N} \setminus \{0\})} \models \pi$;

• $\mathcal{M}, (q_i)_{i \in I} \models \pi_1 \mathbin{U} \pi_2$ if and only if there exists $k \in \mathbb{N}$ such that $\mathcal{M}, q_k \models \pi_2$ and $\mathcal{M}, q_j \models \pi_1$ for each $0 \leq j < k$.

2.3.2 LTL and CTL

Two useful CTL*’s sub-logics are CTL and LTL. The former is a branching time logic, while the latter is a linear time logic. The difference between these classes is related with how they handle alternatives paths. In particular, in branching time logic it is possible to quantify over paths starting at a given state, while in linear time logic it is possible to describe properties holding along a single path only.

Linear Temporal Logic, or LTL, (see [117, 118]) defines properties of paths as opposed to state’s properties. In particular, every LTL formula has the form $A \pi$ where $\pi$ is a path formula whose state sub-formulæ are propositional symbols.

**Definition 2.3.3 (LTL - Syntax)** Let $S$ be a set of propositional symbols and $P \in S$. State formulæ of LTL over $S$ are defined by the following grammar:

$$\varphi ::= A \pi$$

where a formula $\pi$ is defined as:

$$\pi ::= \text{tt} | P | \neg \pi | \pi \lor \pi | \circ \pi | \pi \mathbin{U} \pi$$

Computation Tree Logic, or CTL, (see [24, 25]) consists of those CTL* formulæ in which each temporal operator immediately follows a path quantifier.

**Definition 2.3.4 (CTL - Syntax)** Let $S$ be a set of propositional symbols and $P \in S$. State formulæ of CTL over $S$ are defined by the following grammar:

$$\varphi ::= \text{tt} | P | \neg \varphi | \varphi \lor \varphi | A \pi$$

where $\pi$ is a CTL path formula and it is defined by:

$$\pi ::= \text{tt} | \neg \pi | \circ \varphi | \varphi \mathbin{U} \varphi$$

It has been proved in [93, 62, 46] that the three logics CTL*, CTL, and LTL have different expressive power. In particular, the property expressed by the LTL formula $A \diamond p$ cannot be expressed in CTL. In the same way, there is no LTL formula that is equivalent to the CTL formula $A \square E p$. Thus, the CTL formula $A \diamond p \lor A \square p$ cannot be expressed by either CTL or LTL.
2.4 Dense Temporal Logics

In this section, we present some dense time temporal logics. First, we need to introduce the concept of d-path. D-paths will replace discrete paths in the semantics of dense temporal logics and generalise the concept of p-path presented in [1].

**Definition 2.4.1 (D-Path)** Let \( M = \langle \mathcal{Q}, \Delta_C, \Delta_D \rightarrow, \Lambda \rangle \) be a model and \( I \subseteq \mathbb{N} \) be an initial segment of natural numbers. Moreover, let \( (\rho_i)_{i \in I} \) be a family of functions such that \( \rho_i : \text{Dom}(\rho_i) \rightarrow \mathcal{Q} \) and, if \( j \) is the maximum in \( I \), then \( \text{Dom}(\rho_j) = \mathbb{R}_{\geq 0} \), otherwise \( \text{Dom}(\rho_i) = [0, t_j] \) with \( t_j \in \mathbb{R}_{\geq 0} \). We say that \( (\rho_i)_{i \in I} \) is a \( M \)'s d-path from \( q \in \mathcal{Q} \) if:

- \( \rho_0(0) = q \);
- for all \( i \in I \) and all \( t \in \text{Dom}(\rho_i) \) there exists a \( \lambda \in \Delta_C \) such that, \( \rho_i(0) \xrightarrow{\lambda} \rho_i(t) \);
- for all \( i \in I \setminus \{0\} \) there exists \( \epsilon \in \Delta_D \) such that \( \rho_{i-1}(t_{i-1}) \xleftarrow{\epsilon} \rho_i(0) \).

**Example 2.4.2** Let us consider the model \( M = \langle \mathbb{R}^2, \mathbb{R}, \{\epsilon\} \rightarrow, \Lambda \rangle \) where \( \Lambda(q) = \text{tt} \) for any \( q \in \mathbb{R}^2 \), \( (s_1, s_2) \xrightarrow{\lambda} (s_1 + \lambda, \cos s_2 + \lambda) \) for all \( \lambda \in \mathbb{R} \), and \( (s_1, s_2) \xrightarrow{\epsilon} (0, s_2) \). In such case, \( (\rho_i)_{i \in \{0\}} \), with \( \rho_0(t) = (t, \cos t) \) and \( \text{Dom}(\rho_0) = \mathbb{R}_{\geq 0} \), \( (\rho_i')_{i \in \mathbb{N}} \), with \( \rho_i'(t) = (t, \cos t) \) and \( \text{Dom}(\rho_i') = [0, 2\pi i] \), and \( (\rho_i'')_{i \in \mathbb{N}} \), with

\[
\rho_i''(t) = \begin{cases} (t, \cos t) & \text{if } i \text{ is even} \\ (-t, \cos -t) & \text{otherwise} \end{cases}
\]

and \( \text{Dom}(\rho_i'') = [0, 2\pi i] \), are \( M \)'s d-paths from \( (0, 0) \).

As noticed above, the notion of d-path generalise s-path. As a matter of fact, s-paths specify evolutions of systems which reaches at most one state at every time instant. On the contrary, d-paths can describe evolutions reaching many states at the same time.

Since for all \( i \in I \) and all \( t \in \text{Dom}(\rho_i) \) there exists a \( \lambda \in \Delta_C \) such that \( \rho_i(0) \xrightarrow{\lambda} \rho_i(t') \), the operator \( \circ \) does not have much sense in this context. For this reason, we avoid this operator in giving a semantics on dense structures. In the following sections, we present three logics, CTL*_{\sim \circ}, CTL_{\sim \circ}, \) and LTL_{\sim \circ}, which are dense time versions of CTL*, CTL, and LTL, respectively.

### 2.4.1 CTL* without the \( \circ \) Operator

**Definition 2.4.3 (CTL*_{\sim \circ} - Syntax)** Let \( \mathcal{S} \) be a set of propositional symbols and \( P \in \mathcal{S} \). CTL*_{\sim \circ}'s state formulae are syntactically derived as:

\[
\varphi ::= \text{tt} \mid P \mid \neg \varphi \mid \varphi \lor \varphi \mid \mathbf{A} \varphi
\]

where \( \pi \) is a CTL*_{\sim \circ} path formula and it is defined by:

\[
\pi ::= \varphi \mid \neg \pi \mid \pi \lor \pi \mid \pi \mathbf{U} \pi
\]
2. Models and Temporal Logics

Definition 2.4.4 (CTL*\textsubscript{−◦} - Semantics) Let 
\(M\) be a timed model and \(q\) be a state of \(\mathcal{M}\). We say that \(q\) satisfies the CTL*\textsubscript{−◦} state formula \(\varphi\) on \(\mathcal{M}\), denoted by \(M, q \vdash \varphi\) as follows:

- \(M, q \vdash \top\) always holds;
- \(M, q \vdash \varphi\) if and only if \(\varphi \in \Lambda(q)\);
- \(M, q \vdash \neg \varphi\) if and only if \(q \not\models \varphi\) does not hold;
- \(M, q \vdash \varphi_1 \lor \varphi_2\) if and only if \(M, q \models \varphi_1\) or \(M, q \models \varphi_2\);
- \(M, q \vdash A \pi\) if and only if \(M, (\rho_i)_{i \in I} \models \pi\) for all d-path \((\rho_i)_{i \in I}\) from \(q\).

Moreover, if \((\rho_i)_{i \in I}\) is a d-path of \(\mathcal{M}\), we say that \((\rho_i)_{i \in I}\) satisfies the CTL*\textsubscript{−◦} path formula \(\pi\) on \(\mathcal{M}\), denoted by \(M, (\rho_i)_{i \in I} \models \pi\) as follows:

- \(M, (\rho_i)_{i \in I} \models \varphi\) if and only if \(M, \rho_0(0) \models \varphi\);
- \(M, (\rho_i)_{i \in I} \models \neg \pi\) if and only if \(M, (\rho_i)_{i \in I} \models \pi\) does not hold;
- \(M, (\rho_i)_{i \in I} \models \pi_1 \lor \pi_2\) if and only if \(M, (\rho_i)_{i \in I} \models \pi_1\) or \(M, (\rho_i)_{i \in I} \models \pi_2\);
- \(M, (\rho_i)_{i \in I} \models \pi_1 U \pi_2\) if and only if
  - for all \(i \in I\) and \(t \in \text{Dom}(i)\), if \(M, \rho_i(t) \models \neg \pi_1\), then either there exists a \(t' \in [0,t]\) such that \(M, \rho_i(t') \models \pi_2\)
  - or there exist a \(j \in [0,i - 1]\) and a \(t' \in \text{Dom}(\rho_j)\) such that \(M, \rho_j(t') \models \pi_2\).

2.4.2 LTL*\textsubscript{−◦} and CTL*\textsubscript{−◦} Logics

Let us introduce here the syntax of LTL*\textsubscript{−◦}, a sub-logic of CTL*\textsubscript{−◦} (see [33]).

Definition 2.4.5 (LTL*\textsubscript{−◦} - Syntax) Let \(S\) be a set of propositional symbols and \(P \in S\). Any formula of LTL*\textsubscript{−◦} over \(S\) is generated by the following grammar:

\[ \varphi ::= A \pi \]

where \(\pi\) is derived as:

\[ \pi ::= \top \mid P \mid \neg \pi \mid \pi \lor \pi \mid \pi U \pi \]

In an analogous manner, we can define the syntax of CTL*\textsubscript{−◦} logic (see [33]).

Definition 2.4.6 (CTL*\textsubscript{−◦} - Syntax) Let \(S\) be a set of propositional symbols and \(P \in S\). Any CTL*\textsubscript{−◦} formula, \(\varphi\), over \(S\) is defined by:

\[ \varphi ::= \top \mid P \mid \neg \varphi \mid \varphi \lor \varphi \mid A \pi \]

where \(\pi\) is derived as:

\[ \pi ::= \top \mid \neg \pi \mid \varphi U \varphi \]
2.4. Dense Temporal Logics

The semantics of both LTL and CTL can be easily derived from the CTL\(^*\)’s semantics. Since the above logics are analogous to LTL and CTL, respectively, we will use the same abbreviations introduced in Section 2.2 for these logics.

2.4.3 Timed CTL Logic

Timed CTL, or simply TCTL, was introduced in [1] to characterise properties over timed models. Let us introduce here the syntax and semantics of TCTL.

Definition 2.4.7 (TCTL - Syntax) Let \( S \) be a set of propositional symbols and \( P \in S \). Any formula of TCTL over \( S \) is generated by the following grammar:

\[
\varphi ::= tt \mid P \mid \neg \varphi \mid \varphi \lor \varphi \mid A(\varphi U \preceq c \varphi) \mid E(\varphi U \preceq c \varphi)
\]

where \( \preceq \in \{\leq, <, =, >, \geq\} \) and \( c \in \mathbb{R} \).

Beside to the usual shorthands we use \( E(\preceq c \varphi) \) meaning \( E(tt U \preceq c \varphi) \), \( A(\preceq c \varphi) \) for \( A(tt U \preceq c \varphi) \), \( E(\preceq c \varphi) \) to denote \( \neg A(\preceq c \varphi) \), and \( A(\preceq c \varphi) \) meaning \( \neg E(\preceq c \varphi) \).

Now we define the TCTL semantics.

Definition 2.4.8 (TCTL - Semantics) Let \( M \) be a timed model, \( q \) be a state of \( M \), and \( \varphi \) be a TCTL formula. The satisfaction relation \( M, q \models \varphi \) is recursively defined as:

- \( M, q \models tt \) always holds;
- \( M, q \models P \) if and only if \( P \in \Lambda_S(q) \);
- \( M, q \models \neg \varphi \) if and only if \( M, q \not\models \varphi \) does not hold;
- \( M, q \models \varphi_1 \lor \varphi_2 \) if and only if \( M, q \models \varphi_1 \) or \( M, q \models \varphi_2 \);

for all \( M \)'s \( d \)-path \( (\rho_i)_{i \in I} \) from \( q \), there exists \( i \in I \), a real value \( t \in Dom(\rho_i) \), and a sequence of times \( (t_j)_{j \in [0,i]} \) such that if \( t_j \in Dom(\rho_j) \) for all \( j \in [0,i] \), \( \rho_i(0) \overset{t}{\rightarrow} \rho_i(t) \), and \( \rho_k(0) \overset{t}{\rightarrow} \rho_k(t_k) \), for all \( k \in [0,i-1] \), then \( \sum_{j \in [0,i]} t_j \leq c \), \( M, \rho_i(t) \models \pi_2 \), \( M, \rho_i(t') \models \pi_1 \) for all \( 0 \leq t' < t \), and \( M, \rho_k(t'') \models \pi_1 \) for all \( t'' \in Dom(\rho_k) \) and all \( k \in [0,i-1] \);
2. Models and Temporal Logics

2.5 Model Checking

Model checking (see [47, 120, 48]) is a formal automatic technique for verifying model's properties. This technique has many advantages over traditional approaches such simulation or testing. In particular, model checking guarantees the correct and complete verification of a property dealing with all the behaviours of the system, whereas simulation or testing consider only some of the possible trajectories. Usually, model’s specifications are written in some propositional temporal logic and the analysis consists in the verification of assertions of type $\mathcal{M}, q \models \psi$, where $\mathcal{M}$ is a model, $q$ is a $\mathcal{M}$’s state, and $\psi$ is the property that we want to verify. If the required property does not hold, the user is provided with a counterexample for it.

Even if, given a model $\mathcal{M}$ and a property $\psi$, verifying $\psi$ on $\mathcal{M}$ is not decidable in general, there exist algorithms which verify CTL, LTL, and CTL* formulae over finite models. In particular, the following result for CTL has been proved in [47].

**Theorem 2.5.1** Let $\psi$ be a CTL formula, $\mathcal{M} = (Q, \rightarrow, \Lambda)$ be a finite model, and $q$ be a $\mathcal{M}$’s state. There exists an algorithm for deciding $\mathcal{M}, q \models \psi$ that runs in time $O(|\psi| \cdot (|Q| + | \rightarrow |))$.

Moreover, [95] describes an algorithm, based on tableaux, which proves the decidability of LTL model checking on finite model. In particular, the following theorem holds.

**Theorem 2.5.2** Let $\psi$ be a LTL formula, $\mathcal{M} = (Q, \rightarrow, \Lambda)$ be a finite model, and $q$ be a $\mathcal{M}$’s state. There exists an algorithm for deciding $\mathcal{M}, q \models \psi$ that runs in time $O(2^{|\psi|} \cdot (|Q| + | \rightarrow |))$. 

There exists $\mathcal{M}$’s path $(\rho_i)_{i \in I}$ from $q$, a natural number $i \in I$, a real value $t \in \text{Dom}(\rho_i)$, and a sequence of times $(t_j)_{j \in [0, i]}$ such that $t_j \in \text{Dom}(\rho_j)$ for all $j \in [0, i]$, $\rho_k(0) \xrightarrow{t_k} \rho_k(t_k)$, for all $k \in [0, i - 1]$, $\sum_{j \in [0, i]} t_j \geq c$, $\mathcal{M}, \rho_i(t) \models \pi_2$, $\mathcal{M}, \rho_i(t') \models \pi_1$ for all $0 \leq t' < t$, and $\mathcal{M}, \rho_k(t'') \models \pi_1$ for all $t'' \in \text{Dom}(\rho_k)$ and all $k \in [0, i - 1]$.

Intuitively, $\mathcal{M}, q \models \bigwedge (\varphi_1 \cup \varphi_2)$ means that, for all d-path $(\rho_i)_{i \in I}$ from $q$, the formula $\varphi_1$ holds until $\varphi_2$ does and the time $t$ needed to reach the first state satisfying $\varphi_2$ in $(\rho_i)_{i \in I}$ is such that the formula $t \geq c$ holds. In an analogous way, $\mathcal{M}, q \models \bigwedge (\varphi_1 \cup \varphi_2)$ means that there exists a d-path $(\rho_i)_{i \in I}$ from $q$ such that the first state satisfying $\varphi_2$ in $(\rho_i)_{i \in I}$ is reachable in time $t$, with $t \geq c$, and $\varphi_1$ holds until $\varphi_2$ does.

Notice that the semantics which we give is different from the semantics proposed in [1]. However, it is easy to prove that they are equivalent in our domain.
Furthermore, it has been proved that the LTL model checking problem for finite models is \textsc{PSPACE}-complete and that the model checking problem for CTL* has the same complexity as the LTL’s one (see [63, 48]).

Usually real systems have many components which can interact with each other. Unfortunately, models describing such systems may have a huge number of states i.e. exponential with respect to the number of system’s components. This problem is known as the \textit{state explosion problem}. Different techniques have been developed in order to reduce model’s states: symbolic model checking [104], abstract model checking [53], partial order reduction [76, 49], equivalence reductions [94], etc. In the following section, we present two central notions in this field also used to reduce models: \textit{simulation} and \textit{bisimulation} relations. These relations help to increase the efficiencies of model checking algorithms allowing to replace large models with smaller models that satisfy almost the same properties.

\section*{2.6 Simulation and Bisimulation}

Many finite models satisfy the same temporal properties even if they are different. For instance, a CTL* formula holds on the model presented in Figure 2.2(a) if and only if it holds on the one of Figure 2.2(b).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figures/figure2.2.png}
\caption{Two different models satisfying the same temporal properties.}
\end{figure}

\subsection*{2.6.1 Simulation}

The notion of \textit{simulation} was introduced by Milner in [106] as a means to compare programs. Roughly, a model $M$ simulates a model $M'$, if every behaviour of $M'$ can be matched by $M$. Notice that $M'$ may contain transitions which are not in $M$.

\begin{definition}[Simulation Relation]
Let $S$ and $S'$ be two sets of propositional symbols and $M = (Q, \Delta_C, \Delta_D, \rightarrow, \Lambda_S)$ and $M' = (Q', \Delta_C, \Delta_D, \rightarrow', \Lambda_{S'})$ be two models. A relation $S \subseteq Q \times Q'$ is a simulation relation between $M$ and $M'$ if and only if, for all $q$ and $q'$, if $\langle q, q' \rangle \in S$ then:

- $\Lambda_S(q) \cap S' = \Lambda_{S'}(q')$;

- for all $\bar{q} \in Q$ and $\sigma \in \mathcal{E}$ such that $q \xrightarrow{\sigma} \bar{q}$ there exists a $\bar{q}'$ such that $q' \xrightarrow{\sigma'} \bar{q}'$ and $\langle \bar{q}, \bar{q}' \rangle \in S$.

If there exists a simulation relation $S$ between $M$ and $M'$ and $\langle q, q' \rangle \in S$ then we say that $q'$ simulates $q$ through $S$ and we will write $q \preceq_S q'$. Moreover, if there exist
two simulation $S$ and $S'$ such that $q \preceq_S q'$ and $q' \preceq_S q$ then we say that $q$ and $q'$ are simulation equivalent and we write $q \simeq q'$.

By extension, given two models $M = \langle Q, \Delta_C, \Delta_D, \to, \Lambda_S \rangle$ and $M' = \langle Q', \Delta_C, \Delta_D, \to', \Lambda_S' \rangle$, if there exists a simulation $S \subseteq Q \times Q'$ between them, then we say that $M'$ simulates $M$ through $S$ and we write either $M \preceq_S M'$ or $M \not\preceq M'$. Moreover, if $M'$ simulates $M$ and $M$ simulates $M'$, we say that $M$ and $M'$ are simulation equivalent.

It is easy to prove that, for any model $M = \langle Q, \Delta_C, \Delta_D, \to, \Lambda_S \rangle$, the relation $\simeq$ over $Q \times Q$ is an equivalence relation. It follows that there exist the $\simeq$’s equivalence classes $[q]_\simeq = \{q' | q' \simeq q\}$ and the quotient set $Q/\simeq = \{[q]_\simeq | q \in Q\}$. In [35], it has been proved that, given a model $M = \langle Q, \Delta_C, \Delta_D, \to, \Lambda_S \rangle$, there exists an unique transition relation $\rightarrow_\simeq$ such that $\to_\simeq$ is minimal and $M/\simeq = \langle Q/\simeq, \Delta_C, \Delta_D, \to_\simeq, \Lambda_S/\simeq \rangle$, where $\Lambda_S/\simeq ([q]_\simeq) = \bigcup_{q' \in [q]_\simeq} \Lambda_S(q')$, are simulation equivalent.

Moreover, the following result has been proved in [72].

**Theorem 2.6.2** Let $M = \langle Q, \Delta_C, \Delta_D, \to, \Lambda_S \rangle$ and $M' = \langle Q', \Delta_C, \Delta_D, \to', \Lambda_S' \rangle$ be two models such that $M \not\preceq M'$. Then $M', q \models \varphi$ implies $M, q \models \varphi$ for all $q \in Q$ and for all LTL formula $\varphi$ with atomic propositions in $S'$.

Hence, for any model $M$, $M/\simeq$ satisfies an LTL formula $\varphi$ if and only if $M$ does.

### 2.6.2 Bisimulation

The notion of bisimulation has been introduced in many fields of computer science with different purposes (see [133, 113, 65, 108]), but it was first defined in [133] by van Benthem as an equivalence principle between models. A relation $B$ between $M$ and $M'$ is a bisimulation if and only if $M'$ simulates $M$ through $B$ and $M$ simulates $M'$ through $B$’s inverse.

**Definition 2.6.3 (Bisimulation Relation)** Let $S$ be a set of propositional symbols and $M = \langle Q, \Delta_C, \Delta_D, \to, \Lambda_S \rangle$ and $M' = \langle Q', \Delta_C, \Delta_D, \to', \Lambda_S' \rangle$ be two models. A relation $B \subseteq Q \times Q'$ is a bisimulation relation between $M$ and $M'$ if and only if:

- $M \preceq_B M'$;
- $M' \preceq_B M$, where $B^{-1} = \{(q', q) | (q, q') \in B\}$.

If there exists a bisimulation relation $B$ between $M$ and $M'$ and $(q, q') \in B$ then we say that $q$ and $q'$ are bisimilar and we write $q \equiv q'$.

By extension, if there exists a bisimulation relation $B$ between $M$ and $M'$, then we say that $M$ and $M'$ are bisimilar and we write $M \equiv M'$.

Notice that not all bisimulations are equivalence relations. However it is easy to prove that the reflexive, symmetric, and transitive closure of any bisimulation is a bisimulation. The proof of the following lemma can be found in [107].
Lemma 2.6.4 Let $M = (Q, \Delta_C, \Delta_D, \rightarrow, \Lambda)$ be a model. The relation $\equiv$ over $Q \times Q$ is both an equivalence relation and a bisimulation between $M$ and itself. Moreover, it is the maximal bisimulation, thus if $B$ is a bisimulation between $M$ and itself, then $B \subseteq \equiv$.

Since, given a model $M = (Q, \Delta_C, \Delta_D, \rightarrow, \Lambda)$, $\equiv$ over $Q \times Q$ is an equivalence relation, there exist both $\equiv$’s equivalence classes $[q]_{\equiv} = \{ q' \mid q' \equiv q \}$ and quotient $Q/\equiv = \{ [q]_{\equiv} \mid q \in Q \}$. Hence, for each model $M$, we can define the model $M/\equiv = (Q/\equiv, \Delta_C, \Delta_D, \rightarrow', \Lambda')$, where $\Lambda'(\{[q]_{\equiv}\}) = \Lambda(q)$ and the relation $\rightarrow'$ is defined as $\rightarrow' = \{ ([q]_{\equiv}, \sigma, [q']_{\equiv}) \mid (q, \sigma, q') \in \rightarrow \}$. For such model, the following result holds.

**Theorem 2.6.5** The model $M/\equiv$ is the smallest model bisimilar to $M$, with respect to states and arcs.

From now on, we say that $M/\equiv$ is the bisimulation quotient of $M$. The following result has been proved in [34].

**Theorem 2.6.6** Let $M = (Q, \Delta_C, \Delta_D, \rightarrow, \Lambda)$ and $M' = (Q', \Delta_C, \Delta_D, \rightarrow', \Lambda')$ be two timed models such that $M \equiv M'$. Then, for all CTL* formula $\varphi$ and for all $q \in Q$, $M, q \models \varphi$ holds if and only if $M', q \models \varphi$ holds.

It follows that for any model $M$, $M/\equiv$ satisfies an CTL* formula $\varphi$ if and only if $M \models \varphi$. Moreover, the following result holds for TCTL (see [132]).

**Theorem 2.6.7** Let $M = (Q, \mathbb{R}_{\geq 0}, \Delta_D, \rightarrow, \Lambda)$ and $M' = (Q', \mathbb{R}_{\geq 0}, \Delta_D, \rightarrow', \Lambda')$ be two timed models such that $M \equiv M'$. Then, for all TCTL formula $\varphi$ and for all $q \in Q, M, q \models \varphi$ holds if and only if $M', q \models \varphi$ holds.

The above observation will be exploited in the following chapters, when, to decide a formula over a timed model, we evaluate either its simulation or bisimulation quotient.

**Example 2.6.8** Consider the models $M_1 = (Q_1, \rightarrow_1, \Lambda_1')$ and $M_2 = (Q_2, \rightarrow_2, \Lambda_2')$, where the set of model’s state $Q_1$ is $Q_1 = \{q_1, q_2, q_3\}$, the relation $\rightarrow_1$ is defined as $\rightarrow_1 = \{(q_1, q_2), (q_2, q_3), (q_2, q_2), (q_2, q_3)\}$, the labeling function $\Lambda_1'$ is such that $\Lambda_1'(q_1) = \{A\}, \Lambda_1'(q_2) = \{B\}, \Lambda_1'(q_3) = \{C\}$. The set of model’s state $Q_2$ is $Q_2 = \{q_1, q_2, q_2, q_3\}$, the relation $\rightarrow_2$ is defined as $\rightarrow_2 = \{(q_1, q_2), (q_2, q_3), (q_2, q_2), (q_2, q_3), (q_1, q_2), (q_2, q_2), (q_2, q_2)\}$, and the labeling function $\Lambda_2'$ is such that $\Lambda_2'(q_1) = \{A\}, \Lambda_2'(q_2) = \Lambda_2'(q_2) = \{B\}$ and $\Lambda_2'(q_3) = \{C\}$.

It is easy to see that $M_1$ simulates $M_2$ through $S_1 = \{(q_1, q_1), (q_2, q_2), (q_3, q_3)\}$ while $M_2$ simulates $M_1$ through $S_2 = \{(q_1, q_1), (q_2, q_2), (q_2, q_2), (q_3, q_3)\}$, $M_1 \simeq M_2$. Notice that $M_1$ and $M_2$ are not bisimilar. ■

Many algorithm have been developed to evaluate both simulation and bisimulation quotients of a given model (see [29, 30, 60, 67, 35, 61]). If $M$ is a model, either finite or infinite, and if $M/\equiv$, $\Delta_C$ and $\Delta_D$ are finite, then the following algorithm computes $M/\equiv$. First, we need to introduce the operator $\text{Pre}_\sigma(\cdot) : 2^2 \rightarrow 2^2$, where $\sigma \in \Delta$.

Given a model $(Q, \Delta_C, \Delta_D, \rightarrow, \Lambda)$, $\text{Pre}_\sigma(\cdot)$ is defined as $\text{Pre}_\sigma(P) \overset{\text{def}}{=} \{ p \mid q \in P \land p \xrightarrow{\sigma} q \}$ for all $P \subseteq Q$ and for each $\sigma \in \Delta$. 


Algorithm 1 Bisimulation Algorithm

Require: A model $\mathcal{M} = (Q, \Delta_C, \Delta_D, \rightarrow, \Lambda_S)$ with $\Delta_C, \Delta_D$, and $\mathcal{M}/ \equiv$ finite.
Ensure: Return $\mathcal{M}/ \equiv$.

- $Q/ \equiv \leftarrow \text{compute_initial_partition_from}(S)$
- $(\rightarrow_\equiv) \leftarrow \emptyset$
- while $\exists P, P' \in Q/ \equiv$ and $\exists \sigma \in \Delta$ such that $\emptyset \neq P \cap \text{Pre}_\sigma(P') \neq P$ do
  - $P_1 \leftarrow P \cap \text{Pre}_\sigma(P')$
  - $P_2 \leftarrow P \setminus \text{Pre}_\sigma(P')$
  - $Q/ \equiv \leftarrow (Q/ \equiv \setminus \{P\}) \cup \{P_1, P_2\}$
- end while
- for all $P, P' \in Q/ \equiv$ and $\sigma \in \Delta$ such that $P \subseteq \text{Pre}_\sigma(P')$ do
  - $(\rightarrow_\equiv) \leftarrow (\rightarrow_\equiv) \cup (P, \sigma, P')$
- end for
- for all $P \in Q/ \equiv$ do
  - $\Lambda_S/ \equiv (P) \leftarrow \Lambda_S(p \in P)$
- end for
- Return $(Q/ \equiv, \Delta_C, \Delta_D, \rightarrow_\equiv, \Lambda_S/ \equiv)$
The notion of *hybrid automata* was first introduced in [102, 4] as a model and specification language for hybrid systems, i.e., systems consisting of a discrete program within a continuously changing environment. In the following subsections we introduce formally both syntax and semantics of such formalism and we present the notation used in this thesis.

### 3.1 Syntax

First, we introduce some notations and conventions. To ease the notation, if $p = (p_1, \ldots, p_k)$ and $s = (s_1, \ldots, s_k)$ are vectors in $\mathbb{R}^k$, $r \in \mathbb{R}_{\geq 0}$, $\oplus \in \{-, +\}$, and $\preceq \in \{\leq, <, =, >, \geq\}$, then we will use $p \oplus s$ to denote the vector $(p_1 \oplus s_1, \ldots, p_k \oplus s_k)$ and $\|s\| \preceq r$ to indicate the formula $(s_1 + \ldots + s_k) \preceq r^2$. We use $\sum_{i=1}^{f} s_i$ in place of $s_i + \ldots + s_f$ and capital letters $Z_m$, $Z'_m$, and $Z^n_m$, where $n, m \in \mathbb{N}$, denote variables ranging over $\mathbb{R}$ and $Z$ denotes the vector of variables $(Z_1, \ldots, Z_k)$. In an analogous manner, $Z'$ denotes the vector $(Z'_1, \ldots, Z'_k)$ and $Z^n$ denotes the vector $(Z^n_1, \ldots, Z^n_k)$. If $Z = (Z_1, \ldots, Z_m)$ is a vector of variables, $\Gamma_Z$ denotes the set $\{Z_1, \ldots, Z_m\}$, while, given a variable set $S$, we use the notation $Z_S$ to indicate a vector such that $\Gamma_{Z_S} = S$ (e.g., if $S = \{Z_1, Z_2\}$, then $Z_S$ may indicate either $(Z_1, Z_2)$ or $(Z_2, Z_1)$). The temporal variables $T$ and $T'$ model time and range over $\mathbb{R}_{\geq 0}$. In the following, given a formula $\psi[Z]$, we will denote the set of values satisfying $\psi$ as $\text{Sat}(\psi)$, i.e., $\text{Sat}(\psi) \overset{\text{def}}{=} \{p \mid \psi[p]\}$.

We are now ready to formally introduce hybrid automata. For each state of a discrete automaton we have an invariant condition and a dynamic law. This dynamic law may depend on the initial conditions, i.e., on the values of the continuous variables.
at the beginning of the evolution in the state. The jumps from one discrete state to another are regulated by the activation and reset conditions.

**Definition 3.1.1 (Hybrid Automaton)** Let $\mathcal{L}$ be a first-order language over reals and $\text{Inv}$, $\text{Dyn}$, $\text{Act}$ and $\text{Reset}$ be formulæ of $\mathcal{L}$. A hybrid automaton $H = (\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{V}, \mathcal{E}, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset})$ of dimension $k$ consists of the following components:

1. $\mathcal{Z} = (Z_1, \ldots, Z_k)$ and $\mathcal{Z}^\prime = (Z'_1, \ldots, Z'_k)$ are two vectors of variables ranging over the reals;

2. $(\mathcal{V}, \mathcal{E})$ is a finite directed graph; the vertices in $\mathcal{V}$ are called locations, or control modes, the directed edges in $\mathcal{E}$ are called edges, or control switches;

3. Each vertex $v \in \mathcal{V}$ is labeled by the two formulæ $\text{Inv}(v)[\mathcal{Z}]$ and $\text{Dyn}(v)[\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{T}]$ such that if $\text{Inv}(v)[p]$ is true then $\text{Dyn}(v)[p, p, 0]$ is true; moreover, we define $\text{InvSet} \overset{\text{def}}{=} \{\text{Inv}(v)[Z] \mid v \in \mathcal{V}\}$ and $\text{DynSet} \overset{\text{def}}{=} \{\text{Dyn}(v)[\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{T}] \mid v \in \mathcal{V}\}$;

4. Each edge $e \in \mathcal{E}$ is labeled by the two formulæ $\text{Act}(e)[\mathcal{Z}]$ and $\text{Reset}(e)[\mathcal{Z}, \mathcal{Z}^\prime]$; from such formulæ, we can define the formula $\overline{\text{Reset}(e)[\mathcal{Z}]} \overset{\text{def}}{=} \exists \mathcal{Z}' \text{Inv}(v)[\mathcal{Z}'] \land \text{Act}(e)[\mathcal{Z}'] \land \text{Reset}(e)[\mathcal{Z}', \mathcal{Z}] \land \text{Inv}(w)[\mathcal{Z}]$, where $e = \langle v, w \rangle$, and both the sets $\text{ActSet} \overset{\text{def}}{=} \{\text{Act}(e)[\mathcal{Z}] \mid e \in \mathcal{E}\}$ and $\text{ResetSet} \overset{\text{def}}{=} \{\text{Reset}(e)[\mathcal{Z}, \mathcal{Z}'] \mid e \in \mathcal{E}\}$.

The formulæ $\text{Inv}(v)[\mathcal{Z}]$ and $\text{Dyn}(v)[\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{T}]$ are said invariant of $v$ and dynamics of $v$, respectively, while $\text{Act}(e)[\mathcal{Z}]$ and $\text{Reset}(e)[\mathcal{Z}, \mathcal{Z}^\prime]$ are called activation of $e$ and reset of $e$, respectively. Moreover, if a reset does not depend on $\mathcal{Z}$ then such reset is said to be a constant reset.

By an abuse of notation, in the rest of the thesis, we will often write $\mathcal{I}(v)$, $\mathcal{A}(e)$, and $\mathcal{R}(e)$ to mean $\text{Sat}(\text{Inv}(v))$, $\text{Sat}(\text{Act}(e))$, and $\text{Sat}(\text{Reset}(e))$, respectively.

In our definitions, instead of the classical approach of using differential equations to define the flow, we use the formulæ in $\text{DynSet}$ to describe the continuous evolution without using derivatives. Our approach is similar to the one followed in [32]. For instance, in [90], even though the automata are defined with differential equations, it is necessary to compute their solutions in order to apply the bisimulation algorithm, and express these solutions by $\text{Dyn}(v)[\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{T}]$, whose intuitive meaning is that from $\mathcal{Z}$ after $\mathcal{T}$ instants the continuous flow can reach $\mathcal{Z}^\prime$. Thus, our hybrid automata generalise several recently discovered notions in the hybrid systems theory. Note, as an example, that $O$-minimal hybrid automata [90, 32] are a special case of our hybrid automata, since we do not impose restrictions on the formulæ and on the resets. Moreover, we admit an infinite number of flows, which can also be self-intersecting. Similarly, rectangular hybrid automata [119, 81, 87] can be easily mapped into a subclass of our definition.

In general, we may express hybrid automaton dynamics using differential expressions (either equations or inclusions). Let $\mathcal{F}$ and $\mathcal{R}$ be two functions assigning to each vertex $v \in \mathcal{V}$ a differential equation and a differential inclusion, respectively. We use the notation $H = (\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{V}, \mathcal{E}, \text{Inv}, \mathcal{F}, \text{Act}, \text{Reset})$ instead of $H = (\mathcal{Z}, \mathcal{Z}^\prime, \mathcal{V}, \mathcal{E}, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset})$ to denote the fact that, for each vertex $v \in \mathcal{V}$, the formula
Dyn\(\langle Z, Z', T \rangle\) corresponds to the solution of the differential equation \(f(v)\) when the starting point is \(Z\). In particular, we use \(f_v : \mathbb{R}^k \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^k\) to indicate the solution of \(f(v)\) and \(Dyn(v)|\langle Z, Z', T \rangle|\) is \(Z' = f_v(Z, T)\). In the same way, we use the notation \(H = \langle Z, Z', V, \mathcal{E}, Inv, \mathcal{R}, Act, Reset \rangle\) instead of \(H = \langle Z, Z', V, \mathcal{E}, Inv, Dyn, Act, Reset \rangle\) to denote the fact that, for each vertex \(v \in V\), the formula \(Dyn(v)|\langle Z, Z', T \rangle|\) corresponds to the solution of the differential inclusion \(\mathcal{R}(v)\) when the starting point is \(Z\). In particular, we use \(r_v : \mathbb{R}^k \times \mathbb{R}_{\geq 0} \mapsto 2^{\mathbb{R}^k}\) to indicate the solution of \(\mathcal{R}(v)\) and \(Dyn(v)|\langle Z, Z', T \rangle|\) is equivalent to \(Z' \in r_v(Z, T)\). Notice that both the solutions of \(f\) and \(\mathcal{R}\) may be not computable.

In the remaining part of the section, we introduce the concept of transitive dynamics.

**Definition 3.1.2 (Transitive Formula)** Let \(\psi(Z, Z', T)\) be a first-order formula. Such formula is said transitive if, for all \(\psi[p, r, t]\) and for all \(t_1, t_2 \in \mathbb{R}_{\geq 0}\) such that \(\psi[p, r, t_1]\) and \(\psi[r, s, t_2]\) hold, \(\psi[p, s, t_1 + t_2]\) holds as well.

Hence a transitive dynamics is simply a dynamics formula which is transitive.

**Definition 3.1.3 (Transitive Dynamics)** Let \(H = \langle Z, Z', V, \mathcal{E}, Inv, Dyn, Act, Reset \rangle\) be a hybrid automaton. If the formula \(Dyn(v)|\langle Z, Z', T \rangle|\) transitive, we say that \(Dyn\) is a dynamics transitive on \(v\). Moreover if \(Dyn(v)|\langle Z, Z', T \rangle|\) is transitive for all \(v \in V\), then we say that \(Dyn\) is a transitive dynamics.

**Example 3.1.4** Consider the dynamics \(Dyn(v)|\langle Z, Z', T \rangle|\equiv Z' = 4 \ast T + Z\). Such dynamics is transitive on \(v\). As a matter of the fact, if \(Dyn(v)|\langle p, q, t \rangle|\) and \(Dyn(v)|\langle q, s, t' \rangle|\) hold, then \(q = 4 \ast t + p\) and \(s = 4 \ast t' + q\) by definition. Hence \(s = 4 \ast t' + (4 \ast t + p) = 4 \ast (t' + t) + p\) and thus \(Dyn(v)|\langle p, s, t + t' \rangle|\) holds.

### 3.2 Semantics and Reachability

To formalise the semantics of hybrid automata, we first need to introduce the concept of hybrid automaton’s state.

**Definition 3.2.1 (States and Regions)** Let \(H\) be a hybrid automaton of dimension \(k\). A state \(q\) of \(H\) is a pair \(\langle v, r \rangle\), where \(v \in V\) is a location and \(r = (r_1, \ldots, r_k) \in \mathbb{R}^k\) is an assignment of values for the variables of \(Z\). A state \(\langle v, r \rangle\) is said to be admissible if \(Inv(v)[r]\) is true.

A region \(R\) of \(H\) is a set of states such that all the states are in same location. The region \(R = \{ \langle v, r \rangle \mid \varphi[r] \}\) of the states in location \(v\) whose continuous value satisfies \(\varphi\) is denoted by \(\langle v, \varphi \rangle\).

Intuitively, an execution of a hybrid automaton corresponds to a sequence of transitions from an automaton’s state to another state. Hybrid automata have two kinds of transitions: continuous reachability transition relations, capturing the continuous
evolution of a state according to both formulæ $\text{Dyn}(v)$ and $\text{Inv}(v)$, and discrete reachability transition relation, capturing changes of location driven by the formulæ $\text{Reset}(e)$ and $\text{Act}(e)$.

More formally, we can define hybrid automaton semantics as follow.

Definition 3.2.2 (Hybrid Automaton - Semantics) Let $H$ be a hybrid automaton of dimension $k$. The continuous reachability transition relations $\xrightarrow{\text{C}}$ between admissible states is defined as follows:

$$\langle v, r \rangle \xrightarrow{l} \langle v, s \rangle \iff \exists f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^k \text{ continuous function such that } r = f(0), s = f(t), \text{ and for each } t' \in [0, t] \text{ the formula } \text{Inv}(v)[f(t')] \text{ and } \text{Dyn}(v)[r, f(t'), t'] \text{ hold.}$$

Such function is called flow function.

The discrete reachability transition relation $\xrightarrow{\text{D}}$, where $e \in \mathcal{E}$ between admissible states is defined as follows:

$$\langle v, r \rangle \xrightarrow{(v, u)} \langle v, s \rangle \iff \langle v, u \rangle \xrightarrow{} \text{D} \langle u, s \rangle \iff \text{Act}(\langle v, u \rangle)[r], \text{ Reset}(\langle v, u \rangle)[r, s], \text{ and } \text{Inv}(u)[s] \text{ hold.}$$

We use the notation $q \xrightarrow{\lambda}\ x$ to indicate that either $q \xrightarrow{\lambda}\ q'$, if $\lambda \in \mathbb{R}_{\geq 0}$, or $q \xrightarrow{\lambda}\ q'$, if $\lambda \in \mathcal{E}$. Furthermore, $q \xrightarrow{\lambda}\ q'$ denotes that there exists a $t$ such that $q \xrightarrow{\lambda}\ q'$, while $q \xrightarrow{\lambda}\ q'$, where $e \in \mathcal{E}$, indicates that $q \xrightarrow{\lambda}\ q'$.

Given a hybrid automaton $H$ and a $H$'s region $R = \langle v, \varphi \rangle$, we use $\langle R \rangle_t^c$ and $\langle R \rangle_t^{-c}$ to indicate the set of states reachable from $R$ with a $t'$-time flow where $t' \leq t \in \mathbb{R}_{\geq 0}$, and $\bigcup_{t \in \mathbb{R}_{\geq 0}} \langle R \rangle_t^c$, respectively (i.e., $\langle R \rangle_t^c \overset{\text{def}}{=} \{ q \mid \exists Z \exists 0 \leq T \leq t \langle v, Z \rangle \overset{T}{\xrightarrow{\text{C}}} q \land \varphi[Z] \}$) and $\langle R \rangle_t^{-c} \overset{\text{def}}{=} \{ q \mid \exists Z \exists T \geq 0 \langle v, Z \rangle \overset{T}{\xrightarrow{\text{C}}} q \land \varphi[Z] \}$. Moreover, we use $\langle R \rangle_{t_{e}}^{-c}$ to denote the set of states reachable from $R$ with a discrete transition over an edge $e \in \mathcal{E}$ i.e., $\langle (v, \varphi) \rangle_{(t', v')_{e}}^{-c} \overset{\text{def}}{=} \{ (v', Z') \mid \exists Z, Z' \phi[Z] \land \langle v', Z \rangle \overset{(v', v')_{e}}{\xrightarrow{\text{D}}} (v'', Z') \land v = v' \}$. Finally, we write $\langle R \rangle_{t_{e}}^{-c}$ meaning $\text{Sat}(\varphi)$. Notice that if $R$ is a region, then $\langle R \rangle_{t_{e}}^{-c}$, $\langle R \rangle_{t_{e}}^{-c}$, and $\langle R \rangle_{t_{e}}^{-c}$ are regions as well.

Remark 3.2.3 Some works, for example [78, 99], suggest a semantics which allows the hybrid automaton to “touch” states which do not satisfy invariant if a discrete transition brings the automaton from such states to states satisfying the new invariant. In our view, invariants should be always satisfied as they are conditions sine qua non hybrid evolutions cannot be considered valid. For instance, if we aim to model the temperature of a cooler bringing helium to liquid state, we may use as invariant the formula $\text{Inv}(v)[Z] = Z > 0$. This invariant models the fact that it is not possible to cool an object to 0 Kelvin (see [36, 64]). If we use the semantics used in [78, 99], we are implicitly removing such physics constraint and we admit that the cooler can bring helium to 0 Kelvin. On the contrary, if we use the above semantics such behaviour is not allowed. Moreover, the semantics suggested in [78, 99] allows more hybrid evolutions than our semantics only when the regions satisfying invariants are open. In such case, our semantics captures the same hybrid evolutions by considering the automaton whose invariants are the closures of the original invariants.
3.2. Semantics and Reachability

Example 3.2.4 Let $H = (Z, Z', V, \mathcal{E}, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset})$ be an hybrid automaton with $V \overset{\text{def}}{=} \{v\}$, $\mathcal{E} \overset{\text{def}}{=} \{(v, v)\}$, $\text{Dyn}(v)[Z, Z', T] \overset{\text{def}}{=} Z' = e^T \cdot Z$, $\text{Inv}(v)[Z] \overset{\text{def}}{=} 1 \leq Z < e^2$, $\text{Reset}(e)[Z, Z'] \overset{\text{def}}{=} Z' = 1$, and $\text{Act}(e)[Z] \overset{\text{def}}{=} 4 \leq Z \leq e^2$. Moreover, let $\text{tr}$ be the transition sequence $(v, 1) \xrightarrow{2} (v, e^2) \xrightarrow{(v, e^2)} (v, 1)$. By semantics proposed in [78, 99], $\text{tr}$ is valid, while it is not valid by our semantics. However, if we consider the hybrid automaton $H'$ having the same locations, edges, dynamics, activations, and resets of $H$ and whose invariants are defined by the formula $\text{Inv}(v)[Z] \overset{\text{def}}{=} 1 \leq Z \leq e^2$, then, by our semantics, $\text{tr}$ is a valid sequence for $H'$.

Without loss of generality, we consider only hybrid automata whose formulæ are satisfiable. This assumption is not restrictive, since if this is not the case we can transform the automaton and eliminate the unsatisfiable formulæ. For instance, if there exists an edge $e$ such that $\text{Reset}(e)[Z, Z']$ is unsatisfiable we can simply delete the edge from the automaton.

Using above definition we can introduce the notion of corresponding model.

Definition 3.2.5 (Corresponding Model) Let $H = (Z, Z', V, \mathcal{E}, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset})$ be a hybrid automaton of dimension $k$ and $S = \{P_1[Z], \ldots, P_m[Z]\}$ be a set of atomic propositions whose elements are first-order formulæ over the reals with $k$ free-variables. Moreover, let $\Lambda_S : (V \times \mathbb{R}^k) \mapsto 2^S$ be a labeling function such that $\Lambda_S((v, r)) = \{P_i \in S | P_i[r] \text{ holds}\}$.

The two models $M_{H,S} \overset{\text{def}}{=} (V \times \mathbb{R}^k, \{\lambda_r\}, \mathcal{E}, \rightarrow_U, \Lambda_S)$ and $M'_{H,S} \overset{\text{def}}{=} (V \times \mathbb{R}^k, \mathbb{R}_{>0}, \mathcal{E}, \rightarrow_T, \Lambda_S)$ are called untimed corresponding model and timed corresponding model of $H$, respectively.

Building upon corresponding model, we can introduce the notions of trajectory and trace. A trajectory is a $d$-path of the timed corresponding model, while a trace is a sequence of admissible states.

Definition 3.2.6 (Trajectory and Trace) Let $H$ be a hybrid automaton, $M$ be the timed corresponding model of $H$, and $q$ be an admissible state of $H$. A trajectory of $H$ starting from $q$ is a $M$’s $d$-path $(\rho_i)_{i \in I}$ starting from $q$ such that $\rho_i(0) \overset{\lambda_0}{\rightarrow} \rho_i(t)$ for all $i \in I$ and all $t \in \text{Dom}(\rho_i)$.

Let $J \subseteq \mathbb{N}$ be an initial segment of \(\mathbb{N}\). A trace of $H$ is a sequence $(q_j)_{j \in J}$ of admissible states such that:

1. for all $j \in J \setminus \{0\}$ there exists an edge $e \in \mathcal{E}$ such that either $q_{j-1} \overset{e}{\rightarrow_U} q_j$ or $q_{j-1} \overset{\lambda_0}{\rightarrow} \rho_i q_j$;
2. for all $j \in J \setminus \{0, 1\}$ there exist $c$ and $c'$ in $\mathcal{E}$ such that either $q_{j-2} \overset{c}{\rightarrow_U} q_{j-1} \overset{c'}{\rightarrow_U} q_j$, or $q_{j-2} \overset{c}{\rightarrow_U} q_{j-1} \overset{\lambda_0}{\rightarrow} \rho_i q_j$.

Remark 3.2.7 Condition 2 in the above definition has been introduced to define a notion of hybrid trajectory analogous to the notion of trajectory defined in dynamical
Definition 3.2.8 (Zeno Hybrid Automaton and Trajectory) such that Dom\(\text{Dyn}(v)[Z, Z', T] \equiv Z'_2 = T^2 + Z_2 \wedge Z'_1 = T + Z_1\).

with \(Z = (Z_1, Z_2)\) and \(Z' = (Z'_1, Z'_2)\). By definition of trajectory in dynamical systems and by intuitive meaning of dynamics, there is no trajectory from \((0, 0)\) passing through \(4, 8\), satisfying \(\text{Dyn}\). However, if we use Definition 2 and we remove Condition 2, then \(\langle v, (0, 0) \rangle \xrightarrow{2} \langle v, (2, 4) \rangle \xrightarrow{2} \langle v, (4, 8) \rangle\) from which we could conclude that there is a trajectory from the state \(\langle v, (0, 0) \rangle\) passing through \(\langle v, (4, 8) \rangle\).

Given a trace \((q_i)_{i \in I}\), there can be \(m \in I\) such that \(q_{m-1} \xrightarrow{e_v} q_m \xrightarrow{e} q_{m+1}\). Despite this, if there exists \(\tilde{m} \in I\) such that the transitions \(q_{\tilde{m}-1} \xrightarrow{e_v} q_{\tilde{m}} \xrightarrow{\lambda} q_{\tilde{m}+1}\) hold, then there should exist an edge \(e \in \mathcal{E}\) satisfying either the transition \(q_{\tilde{m}-1} \xrightarrow{e} q_{\tilde{m}}\) or \(q_{\tilde{m}} \xrightarrow{\lambda} q_{\tilde{m}+1}\). Moreover, it is easy to verify that there exists a trace \((q_i)_{i \in \mathbb{N}}\) if and only if there exists a trajectory \((\rho_i)_{i \in I}\) from \(q_0\) passing through \(q_n\).

Notice that there exist trajectories which do not spend much time in continuous evolution and time does not advances on them. Hybrid automata admitting such trajectories are called Zeno hybrid automata.

Definition 3.2.8 (Zeno Hybrid Automaton and Trajectory) Let \(H\) be a hybrid automaton and \((\rho_i)_{i \in \mathbb{N}}\) be a \(H\)'s trajectory. If there exists a real value \(c \in \mathbb{R}^+\) such that \(\text{Dom}(\rho_i) = [0, t_i]\) for all \(i \in \mathbb{N}\) and

\[
\sum_{i=0}^{\infty} t_i \leq c,
\]

then \((\rho_i)_{i \in \mathbb{N}}\) is said to be a Zeno trajectory and \(H\) is said to be a Zeno hybrid automaton.

We can now introduce the formal notion of reachability.

Definition 3.2.9 (Hybrid Automaton - Reachability) Let \(H\) be a hybrid automaton. If there exist \(v, u \in \mathcal{V}\), \(j \in I\), \(t' \in \mathbb{R}_{\geq 0}\) and a \(H\)'s trajectory \((\rho_i)_{i \in I}\), with \(\text{Dom}(\rho_i) = [0, t_i]\), starting in \(\langle v, r \rangle\) such that \(\rho_j(t') = \langle u, s \rangle\) then we say that \(r \in \mathbb{R}^k\) reaches \(s \in \mathbb{R}^k\) in time \(t' + \sum_{i=0}^{j-1} t_i\) in \(H\). We use \(\text{ReachSet}(r)\) to denote the set of points reachable from \(r\). Moreover, given a region \(R \subseteq \mathbb{R}^k\) we use \(\text{ReachSet}(R)\) to denote the set \(\cup_{r \in R} \text{ReachSet}(r)\).

The following result, presented in [3] by Alur et al., suggests an algorithm for computing reachable regions.

Proposition 3.2.10 Let \(H\) be a hybrid automaton and \(R\) be a \(H\)'s region. The reachable region \(\text{ReachSet}([R]_0)\) is least fix point of the equation:

\[
X = \left( R \cup \bigcup_{e \in \mathcal{E}} [X]_e^- \right)^\rightarrow.
\]
3.3. Model Checking for Hybrid Systems

Unfortunately, the least fix point of the above equation is not computable in general.

Notice that there exist \( t \in \mathbb{R}_{\geq 0}, j \in I \), and a trajectory \((q_i)_{i \in \mathbb{N}}\) starting in \( \langle v, r \rangle \) such that \( q_j(t) = \langle u, s \rangle \) for some \( v, u \in \mathcal{V} \), if and only if there exists a trace \((q_i)_{i \in [0,n]}\) of \( H \) such that \( q_0 = \langle v, r \rangle \) and \( q_n = \langle u, s \rangle \), for some \( v, u \in \mathcal{V} \). Hence the following lemma holds.

Lemma 3.2.11 A point \( r \in \mathbb{R}^k \) reaches a point \( s \in \mathbb{R}^k \) in time \( t \) if and only if there exist two locations \( v, u \in \mathcal{V} \) and a trace \((q_i)_{i \in [0,n]}\) of \( H \) such that \( q_0 = \langle v, r \rangle \), \( q_n = \langle u, s \rangle \), and \( t = \sum_{j=1}^n t_j \), where for all \( j \in [1,n] \), \( t_j \) is either zero or such that \( q_{j-1} \xrightarrow{t_j} q_j \).

Given a trace of \( H \) we can identify a path of \( \langle \mathcal{V}, \mathcal{E} \rangle \) as follows.

Definition 3.2.12 (Corresponding Path) Let \( H \) be a hybrid automaton. The corresponding path of a \( H \)'s trace \( tr \) is the path \( ph = (v_i)_{i \in I} \), where \( I \) is an initial interval of natural number, of the graph \( \langle \mathcal{V}, \mathcal{E} \rangle \) obtained by considering the discrete transitions occurring in \( tr \). In this case, we also say that \( ph \) corresponds to \( tr \).

Example 3.2.13 If \( tr = \langle v, r_0 \rangle, \langle v, r_1 \rangle, \langle u, r_2 \rangle, \langle v, r_3 \rangle \), then the corresponding path of \( tr \) is \( ph = \langle v, u, v \rangle \).

The reader should distinguish between corresponding paths of a hybrid automaton \( H \) and paths on \( H \)'s corresponding model. The former are sequences of \( H \)'s locations, while the latter are sequences of \( H \)'s state.

Example 3.2.14 Let us consider the hybrid automaton \( H \), with \( \mathcal{V} \) def \{\( v \)\}, \( \mathcal{E} \) def \{\( (v, v) \)\}, \( Inv(v)[Z] \) def \( Z_1 = 1 \), \( Dyn(v)[Z, Z', T] \) def \( \iff \text{Act}(e)[Z] \) def \( Z_1 = 1 \), and \( Reset(e)[Z, Z'] \) def \( Z_1' = Z_1 \). The transition sequence \( \langle v, (1) \rangle \xrightarrow{e_D} \langle v, (1) \rangle \xrightarrow{e_D} \ldots \xrightarrow{e_D} \langle v, (1) \rangle \) is a valid evolution of \( H \). The trace associated to such sequence is \( tr = \langle v, (1) \rangle, \langle v, (1) \rangle, \ldots, \langle v, (1) \rangle \). It follows that \( \langle \langle v, (1) \rangle, \langle v, (1) \rangle, \ldots, \langle v, (1) \rangle \rangle \) is a path of the untimed corresponding model of \( H \) and \( \langle v, v, \ldots, v \rangle \) is the corresponding path of \( tr \).

3.3 Model Checking for Hybrid Systems

To verify specification on a hybrid automaton, one may want to consider the corresponding model and apply model checking techniques such as the ones mentioned in Section 2.5. Unfortunately, hybrid automata have infinite states and therefore corresponding models are infinite objects. It follows that the standard model checking techniques, which works on finite state models, cannot be applied directly in this context. To solve this problem, many authors suggested the use of equivalence reductions based on relations such as simulation and bisimulation. In particular, since bisimulation guarantees the preservation of branching-time temporal logics such as
30

3. Hybrid Automata

CTL and CTL*, if the bisimulation quotient of a model $M$ is finite, then we may verify CTL and CTL* properties of $M$ applying finite model checking techniques on $M/\sim$. In a similar way, if the $M$'s simulation quotient is finite we may verify LTL properties of $M$ applying finite model checking techniques on $M/\approx$. Obviously, bisimulation has the advantage of preserving more expressive logics, but the abstract structure is required to be so similar to the original one that the reductions allowed are less powerful. On the other hand, simulation preserves less expressive logics, but can reduce to a finite state model a larger class of corresponding models.

Notice that, since from a single hybrid automaton we can get both timed and untimed corresponding models, we can compute simulation on both of them. Analogously, we can compute bisimulation on both the models. For these reasons, we distinguish between the so-called timed-abstract simulations/bisimulations, computed on the untimed corresponding model, and the timed simulation/bisimulation, evaluated on timed corresponding models. When we talk about simulation and bisimulation, we refer to timed-abstract simulation and bisimulation, respectively.

An interesting instance of model checking problem is the verification of safety properties: given a hybrid automaton $H$ and a property $\phi$, we want to test whenever $\phi$ holds along all $H$’s trajectories or not. Since $\phi$ holds along all $H$’s trajectories if and only if there is no reachable state in which $\phi$ does not hold. Hence, verification of safety properties naturally reduces to reachability problem.

Even if it has been proved in [82] that reachability is in general not decidable, many interesting classes of hybrid automata over which reachability is decidable have been characterised in the literature [87, 81, 90, 32]. A common approach in deciding reachability of hybrid automata is that of discretising the automata either using equivalence relations which strongly preserve reachability (e.g., bisimulation [90]) or using abstractions (e.g., predicate abstraction [5, 131]). In the first part of this dissertation, instead we study reachability of hybrid automata translating the reachability problem into first-order formulae over the reals. The formulae we get from the translation include the formulae occurring in the automata and hence we need to know which is the theory used to build the automata.

Definition 3.3.1 (T-Automata) Let $T$ be a theory over the reals. A $T$-automaton $H$ is a hybrid automaton such that the formulae occurring in DynSet, InvSet, ActSet, ResetSet are formulae of $T$.

Moreover, we need to characterise hybrid automata which are defined using first-order formulae.

Definition 3.3.2 (First-Order Automata) We call a hybrid automaton $H$ first-order if the formulae in InvSet, DynSet, ActSet, and ResetSet are first-order formulae over the reals, i.e., $H$ is a $T$-automaton where $T$ is a first-order theory over reals.

3.4 Hybrid Automaton Classes

In this section, we introduce some interesting classes of hybrid automata.
3.4.1 Linear Automata

A very general, but quite interesting class of hybrid automata is the class of linear automata. To formalise such class, we first need to give the definition of both linear formula and linear differential inclusion.

**Definition 3.4.1 (Linear Formula)** Let \( Z \) and \( Z' \) be two variable vectors of dimension \( n \). A linear formula, or polyhedral formula, is a formula over either one or two variable vectors. If it is over \( Z \) and \( Z' \), the linear formula has the form \( A_1 \cdot Z + b_1 \not\preceq_1 Z' \land Z' \not\succeq_2 A_2 \cdot Z + b_2 \), where both \( A_1 \) and \( A_2 \) are matrix of dimension \( n \times n \), both \( b_1 \) and \( b_2 \) are vector of dimension \( n \), and \( \not\preceq_1, \not\succeq_2 \in \{\geq, >\} \). Otherwise, if the linear formula is over \( Z \), then it has the form \( A \cdot Z \preceq b \), where \( A \) is a matrix of dimension \( m \times n \), \( b \) is a vector having dimension \( m \) for some \( m \in \mathbb{N}_{\geq 0} \), and \( \preceq \in \{<, \leq, =, \geq, >\} \).

**Definition 3.4.2 (Linear Differential Inclusion)** A linear differential inclusion is a differential inclusion \( \dot{Z} \in \mathcal{F}(Z) \) which can be expressed by a linear formula over \( Z \) and \( Z' \).

Now, we can give the definition of linear automaton.

**Definition 3.4.3 (Linear Automaton)** A linear automaton is a hybrid automaton such that:

- \( \text{Inv}(v)[Z] \), \( \text{Reset}(e)[Z, Z'] \), and \( \text{Act}(e)[Z] \) are linear formulæ, for each \( v \in V \) and each \( e \in E \);
- \( \text{Dyn}(v)[Z, Z', T] \) can be defined by a linear differential inclusion.

Even if there exist linear automata for which the reachability problem is undecidable, such problem is decidable for many of the linear sub-classes. Moreover, in the case in which the exact reachability cannot be achieved, the requirements on the dynamics can be exploited to perform an approximated reachability analysis (see Part III).

**Multirate Automata**

In [3] Alur et al. introduced multirate automata as an extension of timed automata [8]. Such hybrid automata are characterised by resets which are either the identity or the constant function zero. Moreover, their continuous variables evolve like clocks with rational rates (i.e., \( x \) varies as \( c \cdot t + x \), where \( c \in \mathbb{Z} \), in time \( t \)).

**Definition 3.4.4 (Multirate Automaton)** A multirate automaton is a hybrid automaton such that:

- \( \text{Dyn}(v)[Z, Z', T] \) is the solution of the differential equation \( \dot{Z} = c_v \), where \( c_v = (c_{1,v}, \ldots, c_{k,v}) \) and \( c_{i,v} \in \mathbb{N} \), for each \( i \in [1, k] \) and for each \( v \in V \). The value \( c_{i,v} \) is called rate of \( Z_i \) in \( v \);
for each \( e \in \mathcal{E} \), \( \text{Reset}(e)[Z, Z'] \) is a Boolean conjunction of sub-formulae of the form either \( Z_i' = Z_i \) or \( Z_i' = 0 \);

- for each \( v \in \mathcal{V} \) and for each \( e \in \mathcal{E} \), both \( \text{Inv}(v)[Z] \) and \( \text{Act}(e)[Z] \) are Boolean combinations of inequalities of the form either \( Z_i \overset{\approx}{=} c_k \) or \( Z_i - Z_j \overset{\approx}{=} c_k \) where \( i, j \leq k, k \in \mathbb{N}, c_k \in \mathbb{N} \) and \( \overset{\approx}{=} \in \{<, \leq, =, \geq, >\} \).

As proved in [3], the halting problem for 2-counter machines can be reduced to a reachability problem for 2-rated automata. Exploiting the fact that the 2-counter machines are Turing complete (see [84]), the following result has been proved in [3].

**Theorem 3.4.5** The reachability problem is undecidable for multirate automata.

Despite Theorem 3.4.5, the reachability problem for particular sub-classes of multirate automata is decidable. Consider the following definition.

**Definition 3.4.6 (Simple Hybrid Automaton)** Let \( H \) be a hybrid automaton. If both the formulæ \( \text{Inv}(v)[Z] \) and \( \text{Act}(e)[Z] \) are Boolean combinations of inequalities of the form \( Z_i \overset{\approx}{=} c_k \) and \( Z_i - Z_j \overset{\approx}{=} c_k \) where \( i, j \leq k, k \in \mathbb{N}, c_k \in \mathbb{N} \) and \( \overset{\approx}{=} \in \{<, \leq, =, \geq, >\} \) for all \( v \in \mathcal{V} \), then we say that \( H \) is simple.

In [3], the following theorem is proved.

**Theorem 3.4.7** Simple multirate automata have finite bisimulation quotient.

From the above theorem, we can deduce the decidability of both CTL* model checking and reachability problem on simple multirate automata.

**Rectangular Automata**

Puri and Varaiya in [119] introduced rectangular hybrid automata whose dynamics can be characterised by a differential inclusion of the type \( \dot{x} \in [l, u] \), where \( l \) and \( u \) are rational numbers.

**Definition 3.4.8 (Rectangular Automaton)** A rectangular automaton is a hybrid automaton such that:

- for each \( v \in \mathcal{V} \) and for each \( i \in [1, k] \) there exist \( l_{i,v}, u_{i,v} \in \mathbb{N} \) such that \( l_{i,v} \leq u_{i,v} \) and \( \text{Dyn}(v)[Z, Z', T] \) is a solution for the differential inclusions \( \dot{Z}_i \overset{\approx}{=} l_{i,v} \land \dot{Z}_i \overset{\approx}{=} u_{i,v} \), where \( \overset{\approx}{=} \in \{\geq, >\} \), for all \( i \in [1, k] \);

- for each \( e \in \mathcal{E} \), \( \text{Reset}(e)[Z, Z'] \) is a Boolean conjunction of sub-formulae of the form either \( Z_i' = Z_i \) or \( Z_i' \overset{\approx}{=} Z_i \) where \( \overset{\approx}{=} \in \{\geq, >\} \) and \( l_{i,v}, u_{i,v} \in \mathbb{N} \) is such that \( l_{i,v} \leq u_{i,v} \);

- for each \( v \in \mathcal{V} \) and for each \( i \in [1, k] \) there exist \( \tilde{l}_{i,v}, \tilde{u}_{i,v} \in \mathbb{N} \) such that \( \tilde{l}_{i,v} \leq \tilde{u}_{i,v} \) and \( \text{Inv}(v)[Z] \equiv \bigwedge_{i \in [1,k]} \left( Z_i \overset{\approx}{=} \tilde{l}_{i,v} \land \tilde{u}_{i,v} \overset{\approx}{=} 2 Z_i \right) \) where \( \overset{\approx}{=} \in \{\geq, >\} \);
• for each $e \in V$ and for each $i \in [1, k]$ there exist $\bar{l}_{i,v}, \bar{u}_{i,v} \in \mathbb{N}$ such that $\bar{l}_{i,v} \leq \bar{u}_{i,v}$ and $\text{Act}(e)[Z] \equiv \bigwedge_{i \in [1,k]} \left( Z_i \not\preceq_1 \bar{l}_{i,v} \land \bar{u}_{i,v} \not\preceq_2 Z_i \right)$ where $\not\preceq_1, \not\preceq_2 \in \{\geq, >\}$.

If $\text{Reset}((v, v'))[Z, Z'] \rightarrow Z' = Z_i$ implies $u_{i,v} = u_{i,v'}$ and $l_{i,v} = l_{i,v'}$, then the rectangular automaton is called initialised.

Even if Kopke proved in [87] that the reachability problem for rectangular hybrid automata is in general undecidable, Puri and Varaiya in [119] gave the following result concerning initialised rectangular automata.

**Theorem 3.4.9** The reachability problem for the class of initialised rectangular automata is decidable.

Despite this, it has been proved in [77, 87] that there exist very simple rectangular automata having infinite bisimulation and simulation quotients. In particular, consider the following definitions.

**Definition 3.4.10 (Closed and Uniform Activity)** If the dynamics of a rectangular automaton are solutions for differential equations of the type

$$\dot{Z} \geq l_v \land \dot{Z} \geq u_v,$$

where $l_v = (l_{1,v}, \ldots, l_{k,v})$ and $u_v = (u_{1,v}, \ldots, u_{k,v})$, then such automaton is said to have closed activity.

Furthermore, let $H$ be a rectangular automaton, if $\text{Dyn}(v) \equiv \text{Dyn}(v')$ for all locations $v, v' \in V$, then we say that $H$ has uniform activity.

Henzinger proved in [77] the following theorem.

**Theorem 3.4.11** There exists a 2-dimensional rectangular automaton with uniform activity having infinite bisimulation quotient.

Moreover, Kopke gave in [87] the following result.

**Theorem 3.4.12** There exists a 3-dimensional rectangular automaton with uniform activity having infinite simulation quotient.

Nevertheless, in his dissertation (see [87]), Kopke presented a positive result for 2-dimensional rectangular automaton.

**Theorem 3.4.13** Every 2-dimensional rectangular automaton with closed uniform activity has a finite simulation quotient.

### 3.4.2 O-minimal Automata

Even if multirate and rectangular automata were used to verify properties of many real systems (see [92, 56, 9, 96, 112, 97, 114]), their dynamics are quite simple and cannot represent the continuous evolution of more complex systems. To address this issue, Lafferriere, Pappas and Sastry introduced *O-minimal hybrid automata* in [90]. Such class of hybrid automata guarantees finite bisimulation quotient and, indeed, the decidability for reachability problem and for both CTL and LTL model checking over $\mathcal{T}$-automaton, where $\mathcal{T}$ is a decidable theory.
Definition 3.4.14 (O-Minimal Automaton) Let $H$ be a $T$-hybrid automaton. If $T$ is an O-minimal theory, $H$ has constant resets and, for each location $v$, $H$’s dynamics are of the type $Z' = f_v(Z, T)$, then $H$ is said O-minimal hybrid automaton.

Notice that, since there exists O-minimal dynamics that are not expressible by a differential inclusion (e.g., $\text{Dyn}(v)[Z, Z', T] = \text{tt}$), then not all the O-minimal automata are linear automata.

As reported above, the following result has been proved in [90].

Theorem 3.4.15 O-minimal automata have finite bisimulation quotient.

Since O-minimal automata have finite bisimulation quotient, both reachability problem and CTL* model checking are decidable on any $T$-automaton, with $T$ O-minimal and decidable.
II

Hybrid Automata with Inclusion Dynamics
4 Dynamics and Flow Selections

“A mathematician is a machine for turning coffee into theorems.”

P. Erdős

“A computer scientist is a machine for drinking coffee.”

A. Dovier

As remarked in Chapter 3, we allow the use of first-order formulae, in place of differential equations and inclusions, to define hybrid automaton’s flows. In particular, the dynamics are described through formulae. Since, in general, given a dynamics, we cannot guarantee the existence of a corresponding flow function, in this chapter we introduce and study a set of properties which assure such existence. The conditions we impose on dynamics allow us to use Michael’s selection theorem (see [105, 13]) to translate a reachability problem into a first-order satisfiability problem over the reals.

The novelty of our approach mainly lies in the use of continuous selection results [13] which allow us to consider hybrid automata whose dynamics are non-autonomous inclusions. As a direct consequence, we can derive first-order formulae to encode the reachability problem for hybrid automata.

Notice that all the formulae presented in this and in the following chapters are built upon \( \text{Inv, Dyn, Act, and Reset} \) using standard propositional operators and first-order quantifier. It follows that, if we are considering a \( T \)-automaton, all the presented formulae are definable in \( T \) also. Hence, if \( T \) is decidable, we have the decidability of the problems which are reduced to such formulae.

4.1 Dynamics and Selection Problem

Provided the continuity of \( F \), the existence of a continuous solution for the Cauchy problem

\[
\begin{align*}
\dot{x}(t) &= F(t, x(t)) \\
x(0) &= c
\end{align*}
\]  

(4.1.1)
is ensured by the Cauchy-Kovalevskaya theorem (see [125]). Hence specifying hybrid automaton dynamics through differential equations has the side-effect of guaranteeing the existence of a continuous differentiable flow functions satisfying the dynamics themselves. This observation can be exploited to characterise flows defined by differential equations using first-order formulae [90, 91].

However, as remarked in Chapter 3, we allow the use of formulae, in place of differential equations and inclusions, to define hybrid automaton’s flows. This choice lets us model hybrid automata whose dynamics are not differentiable, but it does not guarantee the existence of a continuous flow function satisfying the dynamics themselves. In particular, given two formulae \( \text{Dyn}(v)[Z, Z', T] \) and \( \text{Inv}(v)[Z] \) specifying the dynamics of a location \( v \) and its invariant, respectively, we are not guaranteed that \( \langle v, p \rangle \xrightarrow{t} C \langle v, q \rangle \) even if for all \( t \in \mathbb{R} \geq 0 \) there exists a \( q_t \in \mathbb{R}^k \) such that \( \text{Dyn}(v)[p, q_t, t] \land \text{Inv}(v)[q_t] \) holds (see Example 4.1.3). Hence, we need to find a set of sufficient conditions for the existence of a continuous function satisfying the dynamics. For this reason, we can formulate the flow specification as a selection problem.

In general, given a family of sets \( \{ S_x : x \in X \} \), a selection, or choice function, is a function \( f : X \mapsto \bigcup_{x \in X} S_x \) such that, for each \( x \in X \), \( f(x) \in S_x \). If \( X \) is finite, then the existence of a selection is obvious. Otherwise, it is guaranteed by the axiom of choice[13]. The reader should notice that the axiom of choice does not guarantee continuity. In particular, there exist families of sets which have no continuous selection.

To find a set of sufficient conditions for the continuity of the selection, we need to introduce the notion of lower semi-continuity (see [13]).

**Definition 4.1.1 (Lower Semi-Continuous Function)** Let \( F : X \mapsto 2^Y \) be a map from \( X \) to \( 2^Y \). We define \( F \) to be lower semi-continuous (abbreviated, l.s.c.) if for each \( x \in X \), for each \( y \in F(x) \), and for each neighbourhood \( U_y \) of \( y \), there exists a neighbourhood \( U_x \) of \( x \) such that for each \( x' \in U_x \) it holds \( F(x') \cap U_y \neq \emptyset \).

Exploiting lower semi-continuity, Michael proved the following result (see [105]).

**Theorem 4.1.2 (Michael’s Selection Theorem)** Let \( X \) be a metric space and \( Y \) be a Banach space. Let \( F \) from \( X \) into the closed convex subsets of \( Y \) be lower semi-continuous. Then there exists a continuous selection \( f : X \mapsto Y \) for \( F \).

The above result gives us a sufficient condition to guarantee the existence of a selection. In particular, if \( F(x) \) is convex and closed for all \( x \in X \), then there exists a selection for \( F \). Notice that to prove Theorem 4.1.2 both closure and convexity of \( F(x) \) are required. As a matter of facts, Example 4.1.3 reports a continuous map from the open interval \((-1, +1)\) into closed and non-convex subsets of \( \mathbb{R}^2 \) which has no continuous selection.

**Example 4.1.3** Consider the map \( \Phi : (-1, +1) \mapsto 2^{\mathbb{R}^2} \) defined as follow:

\[
\Phi(t) \overset{\text{def}}{=} \begin{cases} 
\{ (t \cos \theta, \sin \theta) \mid \frac{1}{2} \leq \theta \leq \frac{1}{2} + 2\pi - |t| \} & \text{if } t \neq 0 \\
\{ (x, y) \mid -1 \leq y \leq 1 \land x = 0 \} & \text{otherwise}
\end{cases}
\]
By definition, if \( t = 0 \), \( \Phi(t) \) is the set of points in the segment between \((0,1)\) and \((0,-1)\). Otherwise, if \( t \neq 0 \), \( \Phi(t) \) is a subset of an ellipsoid in \( \mathbb{R}^2 \) obtained by removing from the ellipsoid itself the section from the angle \( \frac{1}{t} - |t| \) to the angle \( \frac{1}{t} \). Hence, as \( t \) gets smaller, the arc length of removed section decreases, while the removed section itself spins around the origin with increasing angular speed. Moreover, the \( x \)-axis of \( \Phi(t) \) shrinks to zero as \( t \to 0 \), collapsing \( \Phi(t) \) to \( \Phi(0) \).

The function \( \Phi \) can be easily proved to be lower semi-continuous all over \((-1,1)\), while there is no continuous selection defined on the same interval. As a matter of the fact, let us assume by contradiction that there exists a selection \( f(t) \) continuous in \((-1,1)\), then there should exits \( \lim_{t \to 0} f(t) \). But, by definition of \( \Phi \), \( f \) is forced to bounce between \((0,1)\) and \((0,-1)\) as fast as \( t \) gets close to zero. Hence \( \lim_{t \to 0} f(t) \) does not exist and then \( f(t) \) cannot be continuous. Notice that, since \( \Phi(t) \) is not convex for \( t \neq 0 \), \( \Phi \) does not satisfy the hypothesis of Theorem 4.1.2.

In the following sections we exploit Michael’s selection theorem to reduce the reachability problem to a decidability problem for an opportune first-order formula.

### 4.2 Michael’s Form

In this section, we exploit Theorem 4.1.2 and we present a set of conditions for hybrid automata which guarantee the existence of a valid continuous transitions.

First of all, we need to characterise the time instants, at which the automata, starting from a point \( p \) in a location \( v \), can reach a point \( q \), while remaining inside the invariant set of \( v \). Such a characterisation is possible when the automaton is a first-order automaton. We recall that an interval over \( \mathbb{R}_{\geq 0} \) is a set of the form \( \{r \in \mathbb{R}_{\geq 0} \mid a \prec_1 r \prec_2 b\} \), where \( \prec_1, \prec_2 \) are in \( \{<, \leq\} \), \( a \in \mathbb{R}_{\geq 0}, b \in \mathbb{R}_{\geq 0} \cup \{+\infty\} \), and \( a \leq b \).
Lemma 4.2.1 Let $H$ be a first-order hybrid automaton. Let $p \in \mathbb{R}^k$ be such that $\text{Inv}(v)[p]$ holds. The formula $\exists Z'(\text{Dyn}(v)[p, Z', 0] \land \text{Inv}(v)[Z'])$ holds.

Proof. The thesis follows from the fact that $\text{Inv}(v)[p]$ implies $\text{Dyn}(v)[p, p, 0]$.

The above lemma allows us to focus on the interval $I^v_p$ of time instants, for which there are dynamics that start from $p$ and remain inside the invariant of $v$—these dynamics are main objects of our interest.

Definition 4.2.2 ($I^H_{v,p}$ and $F^H_{v,p}$) Let $H = \langle Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle$ be a first-order hybrid automaton. Let $v$ be a location of $H$ and $p$ be such that $\text{Inv}(v)[p]$ holds. $I^H_{v,p}$ is the interval of time instants satisfying the following:

1. the formula $\forall T \in I^H_{v,p} \exists Z'(\text{Dyn}(v)[p, Z', T] \land \text{Inv}(v)[Z'])$ holds;
2. $0 \in I^H_{v,p}$;
3. $I^H_{v,p}$ is maximal with respect to the first two requirements.

Define the function $F^H_{v,p} : I^H_{v,p} \mapsto 2^{\mathbb{R}^k}$ as:

$$F^H_{v,p}(T) \overset{\text{def}}{=} \{ q | \text{Dyn}(v)[p, q, T] \land \text{Inv}(v)[q] \}.$$

We now possess all the ingredients to introduce Michael's Form.

Definition 4.2.3 (Michael's Form) Let $H = \langle Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle$ be a hybrid automaton. We say that $H$ is in Michael's form if:

1. $H$ is a first-order automaton;
2. For each $v \in V$ and for all $p \in T(v)$, the function $F^H_{v,p}$ is lower semi-continuous, and, for each $t \in I^H_{v,p}$, the set $F^H_{v,p}(t)$ is closed and convex.

Condition 2 of Definition 4.2.3 imposes a certain kind of continuity on the set of trajectories. Moreover, it requires that for each point $p$ and for each time instant $t$ the set of points reachable from $p$ at time $t$ is a closed convex set. This condition will allow us to exploit Michael's selection theorem to find valid continuous flows.

Example 4.2.4 Let $H = \langle Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle$ where:

1. $Z = (Z_1, Z_2)$ and $Z' = (Z'_1, Z'_2)$;
2. $V = \{ v \}$ and $E = \{ e \}$, where $e$ goes from $v$ to $v$;
3. $\text{Inv}(v)[Z] \equiv (0 \leq Z_1 \leq 1 \land 0 \leq Z_2 \leq 1)$;
4. $\text{Dyn}(v)[Z, Z', T] \equiv (Z'_1 = T + Z_1 \land Z'_2 \geq T^2 + Z_2)$;
5. $\text{Act}(e)[Z] \equiv (Z_1 = 1 \lor Z_2 = (1 - Z_1)^4)$;
4.2. Michael’s Form

- Reset\((v)[Z, Z']\) \(\equiv (Z'_1 = (Z)^3 + 1 \land Z'_2 = 1)\).

The formulæ in \(H\) are first-order formulæ over the reals. If \(p = (p_1, p_2)\), with \(0 \leq p_1, p_2 \leq 1\), then the function \(F^H_{v, p}\) is defined as \(F^H_{v, p}(t) = \{(q_1, q_2) \mid |q_1 = t + p_1, q_2 \geq t^2 + p_2, \text{ and } 0 \leq q_1 \text{ and } q_2 \leq 1\}\). It is easy to see that \(p \in F^H_{v, p}(0)\) and for each \(t\) the set \(F^H_{v, p}(t)\) is closed and convex, since it is a segment. Moreover, this function is lower semi-continuous over the interval \(I^H_{v, p}\). Hence, \(H\) is in Michael’s form.

We now show how to automatise the identification of a hybrid automaton in Michael’s form. In particular, we present a first-order formula which holds if and only if the considered hybrid automaton is in Michael’s form.

Condition 1 is syntactically verifiable, hence we will focus on Condition 2 of Michael’s form definition. First of all, we need to characterise both \(I^H_{v, p}\) and \(F^H_{v, p}\) by some formulæ. Consider the following formulæ. Consider the following formulæ.

\[
\phi(H, v)[Z, Z', T] \overset{\text{def}}{=} \text{Dym}(v)[Z, Z', T] \land \text{Inv}(v)[Z']
\]

\[
\psi(H, v)[Z, T] \overset{\text{def}}{=} \forall T' (0 \leq T' \leq T \rightarrow (\exists Z' \phi(H, v)[Z, Z', T']))
\]

By \(F^H_{v, p}\) definition, it is easy to prove that \(q \in F^H_{v, p}(t)\) if and only if the formula \(\phi(H, v)[p, q, t]\) holds. Moreover, by \(I^H_{v, p}\) definition, we can deduce as well that \(t \in I^H_{v, p}\) if and only if the formula \(\psi(H, v)[p, t]\) holds.

Now consider the lower semi-continuity requirement. The first-order formula expressing the lower semi-continuity property for \(F^H_{v, Z}\) is the following one.

\[
\text{lsc}(H, v)[Z] \overset{\text{def}}{=} \forall T \forall Z' \left( (\psi(H, v)[Z, T] \land \phi(H, v)[Z, Z', T]) \rightarrow \right.
\]

\[
(\forall T_3 \exists T_3 \forall T' (\|T - T'\| < T_3) \rightarrow (\exists Z'' (\phi(H, v)[Z, Z'', T'] \land \|Z'' - Z'\| < T_3))))
\]

In particular, it is easy to see that \(F^H_{v, p}\) is lower semi-continuous if and only if \(\text{lsc}(H, v)[p]\) holds. Moreover, by convexity and closed set definitions, we can deduce the following formulæ stating that \(F^H_{v, Z}(T)\) is convex and that \(F^H_{v, Z}(T)\) is a closed set, respectively.

\[
\text{Conv}(H, v)[Z, T] \overset{\text{def}}{=} \forall Z', Z'' \left( (\phi(H, v)[Z, Z', T] \land \phi(H, v)[Z, Z'', T]) \rightarrow \right.
\]

\[
(\forall Z_1 0 \leq Z'' \leq 1 \rightarrow \phi(H, v)[Z, Z_0(Z' - Z'') + Z'', T])
\]

\[
\text{Closed}(H, v)[Z, T] \overset{\text{def}}{=} \forall Z' \left( (\forall Z_0 > 0 \exists Z'' \phi(H, v)[Z, Z'', T] \land \right.
\]

\[
\|Z' - Z''\| < Z_0) \rightarrow \phi(H, v)[Z, Z', T])
\]
Finally, by Michael’s form definition, we need a formula which holds if and only if, for all points \( p \) in the invariant and for all times \( t \in I^H_{v,p} \), \( F^H_{v,p}(t) \) is lower semi-continuous and \( F^H_{v,p}(t) \) is closed and convex. Such formula is the formula \( \text{MForm}(H, v) \) defined as:

\[
\text{MForm}(H, v) \overset{\text{def}}{=} \forall Z \ (\text{Inv}(v)[Z] \rightarrow (\forall T \ (\psi(H, v)[Z, T] \rightarrow \text{Conv}(H, v)[Z, T] \land \text{Closed}(H, v)[Z, T]))) \land \text{lsc}(H, v)[Z])
\]

Since any hybrid automaton has a bounded number of locations, we can write the formula:

\[
\bigwedge_{v \in V} \text{MForm}(H, v)
\]

which holds if and only if the corresponding hybrid automaton is in Michael’s form.

Notice that, if \( H \) is a \( T \)-automaton, then \( \bigwedge_{v \in V} \text{MForm}(H, v) \) includes only formulæ in \( T \). It follows that \( \bigwedge_{v \in V} \text{MForm}(H, v) \) is \( T \) as well. Hence, if \( H \) is \( T \)-automaton and \( T \) is decidable, then we can decide whenever \( H \) is in Michael’s form or not.

### 4.3 Reachability

Given a hybrid automaton \( H \) in Michael’s form and a starting region \( R \subseteq \mathbb{R}^k \) characterised by a first-order formula \( \rho \) over the reals, we may wish to compute the region \( \text{ReachSet}_H(R) \subseteq \mathbb{R}^k \) of points that can be reached starting from a point in \( R \) and following a trajectory of \( H \).

Our approach will exploit Michael’s selection theorem for set-valued maps. More specifically, Michael’s selection theorem will guarantee the correctness of a translation into appropriate first-order formulæ of our reachability and model checking problems.

In this section, we demonstrate how the reachability problem over hybrid automata in Michael’s form can be reduced to a first-order satisfiability problem. We start characterising the sets \( I^H_{v,r} \).

**Lemma 4.3.1** Let \( H = (Z, Z', V, \mathcal{E}, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset}) \) be a hybrid automaton in Michael’s form. Consider the first-order formula

\[
Tp(H, v)[Z, T] \overset{\text{def}}{=} \forall 0 \leq T' \leq T \exists Z' (\text{Dyn}(v)[Z, Z', T'] \land \text{Inv}(v)[Z'])
\]

Assume \( v \) to be such that \( \text{Inv}(v)[r] \) holds. It follows that:

\[
t \in I^H_{v,r} \iff Tp(H, v)[r, t] \text{ is true.}
\]

**Proof.** (\( \Rightarrow \)) If \( t \in I^H_{v,r} \), then from definition of \( I^H_{v,r} \), it follows that for each \( t' \in [0, t] \) the formula \( \exists Z' (\text{Dyn}(v)[r, Z', t'] \land \text{Inv}(v)[Z']) \) holds. Hence \( Tp(v)[r, t] \) is true.

(\( \Leftarrow \)) If \( Tp(H, v)[r, t] \) is true, then the formula \( \exists Z' (\text{Dyn}(v)[r, Z', t'] \land \text{Inv}(v)[Z']) \) holds for each \( t' \in [0, t] \), i.e., \( t \in I^H_{v,r} \).
4.3. Reachability

Thanks to Lemma 4.3.1, we can prove the following result.

**Theorem 4.3.2** Let $H = (Z, Z', Y, E, Inv, Dyn, Act, Reset)$ be a hybrid automaton in Michael’s form. Consider the first-order formula

\[
\text{Reach}(H, v)[Z, Z', T] \overset{\text{def}}{=} (T > 0 \land \text{Dyn}(v)[Z, Z', T] \land \text{Tp}(H, v)[Z, T]) \lor (T = 0 \land Z = Z') \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'].
\]

Then following holds:

\[
(v, r) \overset{\text{def}}{\leftrightarrow} (s, t) \text{ is true.}
\]

**Proof.** ($\Rightarrow$) By Definition 3.2.2 we have that:

\[
(v, r) \overset{\text{def}}{\rightarrow} (s, t) \iff \text{there exists } f: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^k \text{ continuous function such that } r = f(0), s = f(t), \text{ and the two formulae } \text{Inv}(v)[f(t')] \text{ and } \text{Dyn}(v)[r, f(t'), t'] \text{ hold for each } t' \in [0, t].
\]

From the fact that for each $t' \in [0, t]$ Dyn(v)[r, f(t'), t'] \land Inv(v)[f(t')] hold, it follows that Tp(H, v)[r, t] holds. Hence we deduce that all the formulae Inv(v)[r], Inv(v)[s], Dyn(v)[r, s, t], and Tp(H, v)[r, t] hold, as stated.

($\Leftarrow$) Let assume that $t = 0$, $r = s$, Inv(v)[r], and Inv(v)[s] holds. Then every continuous function $f$ such that $f(0) = s$ is a valid flow and thus $(v, r) \overset{\text{def}}{\rightarrow} (s, t)$ holds by definition. Now assume that the formulae $t > 0$, Dyn(v)[r, s', t], Tp(H, v)[r, t], Inv(v)[r], Inv(v)[s], and Inv(v)[s] hold. By Lemma 4.3.1 we have that $t \in I_{v, r}^H$. Moreover, $s$ belongs to $F_{v, r}^H(t)$, which is lower semi-continuous with convex and closed images. Consider the function $\tilde{F}: [0, t] \rightarrow 2^{\mathbb{R}^k}$ defined as:

\[
\tilde{F}(T) = \begin{cases} 
\{r\} & \text{if } T = 0 \\
F_{v, r}^H(T) & \text{if } 0 < T < t \\
\{s\} & \text{if } T = t
\end{cases}
\]

It is immediately seen that for each $t'$ in $[0, t]$ $\tilde{F}(t')$ is closed and convex. We prove that $\tilde{F}$ is lower semi-continuous on $[0, t]$. Let $t' \in [0, t]$. We need to consider three distinct cases: (a) $t' = 0$; (b) $0 < t' < t$; (c) $t' = t$.

(a) If $t' = 0$ and $y \in \tilde{F}(0)$, then $y = r$. Let $U_r$ be a neighbourhood of $r$. Since $F_{v, r}^H$ is lower semi-continuous there exists a neighbourhood $U_0$ of 0 in $I_{v, r}^H$, such that for each $t''$ in $U_0$ it holds that $F_{v, r}^H(t'') \cap U_r \neq \emptyset$. Since $[0, t] \subseteq I_{v, r}^H$, we get that $U_0 = U_0 \cap [0, t]$ is a neighbourhood of 0 in $[0, t]$. If $t'' \in U_0$, there are two possible subcases: either $t'' = 0$ or $0 < t'' < t$. If $t'' = 0$, then $\tilde{F}(0) \cap U_r = \{r\} \neq \emptyset$. If, on the other hand, $0 < t'' < t$, then $\tilde{F}(t'') \cap U_r = F_{v, r}^H(t'') \cap U_r \neq \emptyset$.

(b) If $0 < t' < t$ and $y \in \tilde{F}(t')$, then $y \in F_{v, r}^H(t')$. Let $U_y$ be a neighbourhood of $y$. Since $F_{v, r}^H$ is lower semi-continuous, there exists a neighbourhood $U_y$ of $t'$ in $I_{v, r}^H$ such that for each $t''$ in $U_y$ it holds that $F_{v, r}^H(t'') \cap U_y \neq \emptyset$. Since $t' \in (0, t) \subseteq I_{v, r}^H$, we conclude that $U_y = U_y \cap [0, t]$ is a neighbourhood of $t'$ in $[0, t]$. If $t'' \in U_y$, then $\tilde{F}(t'') \cap U_r = F_{v, r}^H(t'') \cap U_r \neq \emptyset$.
(c) If \( t' = t \) and \( y \in \tilde{F}(t) \), then \( y = s \). Let \( U_s \) be a neighbourhood of \( s \). Since \( F^H_{v,r} \) is lower semi-continuous, there exists a neighbourhood \( U_t \) of \( t \) in \( I^H_{v,r} \) such that for each \( t'' \) in \( U_t \), it holds that \( F^H_{v,r}(t'') \cap U_s \neq \emptyset \). Since \([0,t] \subseteq I^H_{v,r}\), we get that \( U'_t = U_t \cap [0,t] \) is a neighbourhood of \( t \) in \([0,t]\). If \( t'' \in U'_t \), then there are two possible sub-cases: namely, either \( t'' = t \) or \( 0 < t'' < t \). If \( t'' = t \), then \( \tilde{F}(0) \cap U_s = \{ s \} \neq \emptyset \). If \( 0 < t'' < t \), then \( \tilde{F}(t'') \cap U_s = F^H_{v,r}(t'') \cap U_s \neq \emptyset \).

Since \( \tilde{F} : [0,t] \mapsto 2^{\mathbb{R}^k} \) is lower semi-continuous, since, for each \( t' \in [0,t] \), \( \tilde{F}(t') \) is closed and convex, and since \([0,t] \) is a metric space, and \( \mathbb{R}^k \) is a Banach space, by Michael’s selection theorem 4.1.2 we may deduce the following: there exists \( f : [0,t] \mapsto \mathbb{R}^k \) continuous selection for \( \tilde{F} \). Hence, by definition of continuous selection (see [13]), \( f \) is a continuous function such that for each \( t' \in [0,t] \) it holds \( f(t') \in \tilde{F}(t') \). From this last statement, we further deduce that: \( f(0) = r; f(t) = s \); for each \( 0 < t' < t \) it holds that \( f(t') \in F^H_{v,r}(t') \), i.e., \( \text{Dyn}(v)[r,f(t'),t'] \) and \( \text{Inv}(v)[f(t')] \). In particular, consider the function \( \tilde{f} : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^k \) defined as:

\[
\tilde{f}(T) = \begin{cases} f(T) & \text{if } T \in [0,t] \\ s & \text{if } T > t \end{cases}
\]

We have demonstrated that \( \tilde{f} \) satisfies all the hypothesis required to conclude that \( (v,r) \xrightarrow{C} (v,s) \), as desired. ■

One may observe that for any edge \( (v,u) \in E \) the discrete reachability formula is characterised by the first-order formula

\[
\text{Reach}(H, (v,u))[Z,Z'] \overset{\text{def}}{=} \text{Inv}(v)[Z] \land \text{Act}((v,u))[Z] \land \\
\text{Reset}((v,u))[Z,Z'] \land \text{Inv}(v)[Z'].
\]

Given a point \( r \in \mathbb{R}^k \), we see that the first-order formula \( \text{Reach}(H,v)[r,Z',t] \), as defined in Theorem 4.3.2, and with free variables in \( Z' \), characterises the set of points reachable from \( r \) at \( v \) using only continuous dynamics. Similarly, the first-order formula \( \text{Reach}(H,e)[r,Z'] \) defines the set of points reachable from \( r \) using the discrete transition \( e \).

Now, suppose that a point \( r \) reaches a point \( s \) in time \( t \) through a trace \( tr \), whose corresponding path is \( ph = (v,u) \). Since, by Definition 3.1.1, \( \text{Dyn}(v)[r,r,0] \) and \( \text{Dyn}(u)[s,s,0] \) hold, we see that \( (v,r) \xrightarrow{C} (v,r) \) and \( (u,s) \xrightarrow{C} (u,s) \). Hence, \( tr \) is equivalent to \( tr' \) of the form \( (v,r) \xrightarrow{C} (v,r_1) \xrightarrow{D} (u,s_1) \xrightarrow{C} (u,s) \) where \( t = t' + t'' \). Thus, the reachability can always be expressed through a trace whose corresponding path is \( ph = (v,u) \) and results in the following first-order formula:

\[
\text{Reach}(H,ph)[Z^0,Z^1,Z^2,Z^3,T] \overset{\text{def}}{=} \exists T_1 \geq 0 \exists T_2 \geq 0 (\text{Reach}(H,v)[Z^0,Z^1,T_1] \land \\
\text{Reach}(H,(v,u))[Z^1,Z^2] \land \\
\text{Reach}(H,u)[Z^2,Z^3,T_2] \land T = T_1 + T_2)
\]
If we have a path \( ph = (v_i)_{i \in [0,h]} \) in the graph \( \langle V, E \rangle \), then following two cases are possible: either it corresponds to a trace of \( H \) or it does not. In both cases, we can express the desired reachability relation with a first-order formula, which characterises all the pairs of \( \mathbb{R}^k \) that can be connected in \( H \) through a trace corresponding to path \( ph = (v_i)_{i \in [0,h]} \):

\[
Reach(H, ph)[Z^0, \ldots, Z^{2h+1}, T] \overset{\text{def}}{=} \exists T_0 \geq 0 \ldots \exists T_h \geq 0 \left( T = \sum_{i=0}^{h} T_i \land Reach(H, v_0)[Z^0, Z^1, T_0] \land \right.
\]

\[
\left. \bigwedge_{i \in [0,h-1]} \left( Reach(H, \{v_i, v_{i+1}\})[Z^{2i+1}, Z^{2i+2}] \land \right. \right.
\]

\[
\left. \left. Reach(H, v_{i+1})[Z^{2i+2}, Z^{2i+3}, T_{i+1}] \right) \right)
\]

Notice that in the above formula we consider only traces in which continuous and discrete transitions are alternating. This is not restrictive since, by reachability and trace definitions, any trace can be mapped into a trace which satisfies the continuous/discrete alternation and has the same starting and finishing states. The following lemma proves that the formula \( Reach(H, ph)[Z^0, \ldots, Z^{2h+1}, T] \) is correct and complete.

**Lemma 4.3.3** Let \( H = \langle Z, Z', V, E, Inv, Dyn, Act, Reset \rangle \) be a hybrid automaton in Michael’s form and \( ph = (v_i)_{i \in [0,h]} \) be a path in \( \langle V, E \rangle \). It holds that \( r \) reaches \( s \) in time \( t \) through a trace \( tr \) whose corresponding path is \( ph \) if and only if \( Reach(H, ph)[r, Z^1, \ldots, Z^{2h}, s, t] \) is satisfiable.

**Proof.**  \( (\Rightarrow) \) Let \( tr = (\ell_i)_{i \in [0,n]} \) with \( \ell_0 = (v_0, r) \) and \( \ell_n = (v_n, s) \). Since, by Definition 3.1.1, \( Dyn(v)[r, r, 0] \) and \( Dyn(u)[s, s, 0] \) hold, if there are two consecutive discrete transitions \( \ell_i \overset{\epsilon}{\rightarrow} D \overset{\epsilon}{\rightarrow} D \ell_{i+1} \overset{\epsilon}{\rightarrow} D \ell_{i+2} \) in \( tr \), we can replace them by \( \ell_i \overset{\epsilon}{\rightarrow} D \ell_{i+1} \overset{\epsilon}{\rightarrow} C \ell_{i+1} \overset{\epsilon}{\rightarrow} D \ell_{i+2} \). Hence, without loss of generality, we may assume that in \( tr \) discrete and continuous transitions are alternated. We may further assume that \( tr \) starts and ends with a continuous transition, since, otherwise, we may simply add either \( \ell_0 \overset{0}{\rightarrow} C \ell_0 \) or \( \ell_n \overset{0}{\rightarrow} C \ell_n \) or both. Hence, without loss of generality, we have that \( n = 2h \). Let \( \ell_i = (v_i, r_i) \) and consider the valuation, which replaces \( Z^i \) by \( r_i \) in the formula \( Reach(H, ph)[r, Z^1, \ldots, Z^{2h}, s, t] \). By induction on \( h \), we can prove that this valuation satisfies \( Reach(H, ph)[r, Z^1, \ldots, Z^{2h}, s, t] \).

\( (\Leftarrow) \) Since \( Reach(H, ph)[r, Z^1, \ldots, Z^{2h}, s, t] \) is satisfiable, there exists an assignment to the \( Z^i \)'s which satisfies it by replacing \( Z^i \) with \( z_i \). Consider the trace \( tr = (\ell_i)_{i \in [0,2h]} \) such that \( \ell_0 = (v, r) \), \( \ell_{2h} = (v_h, s) \), and for each \( i \in [1, h-1] \), we have \( \ell_{2i-1} = (v_{i-1}, z_{2i-1}) \) and \( \ell_{2i} = (v_i, z_{2i}) \). By induction on the length of \( ph \), we can prove that \( tr \) is a trace of \( H \), which connects \( r \) to \( s \) in time \( t \). \( \blacksquare \)
Let $ph$ a path of length $h$. Consider the formula

$$\widehat{\text{Reach}}(H, ph)[Z, Z', T] \overset{\text{def}}{=} \exists Z^1, \ldots, Z^{2h} \text{ Reach}(H, ph)[Z, Z^1, \ldots, Z^{2h}, Z', T].$$

Since $\widehat{\text{Reach}}(H, ph)[r, s, t]$ holds if and only if $\text{Reach}(H, ph)[r, Z^1, \ldots, Z^{2h}, s, t]$ is satisfiable, by Lemma 4.3.3, $r$ reaches $s$ in time $t$ if and only if there exists a path $ph$ of $\langle V, E \rangle$ such that the formula $\widehat{\text{Reach}}(H, ph)[r, s, t]$ holds. So, we could characterise reachability for a hybrid automaton in Michael’s form, considering the disjunction of all the formulæ for all the paths of $\langle V, E \rangle$. Unfortunately, if $\langle V, E \rangle$ has a cycle, then it has an infinite number of paths. In the following chapters, we introduce two classes of hybrid automata in Michael’s form whose valid traces have corresponding paths of bounded length.

### 4.3.1 Reachability for Transitive Dynamics

In this subsection, we give a result concerning reachability for transitive dynamics. Such result will be used in the following chapters to reduce a model checking problem to a decidability problem for a formula over the reals.

**Lemma 4.3.4** Let $H = \langle Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle$ be a hybrid automaton. If $\text{Dyn}$ is a transitive dynamics, then

$$\text{Reach}(H, v)[Z, Z', T] \equiv \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, v)[Z, Z'', T'] \land$$

$$\text{Reach}(H, v)[Z'', Z', T - T'])$$

**Proof.** ($\Rightarrow$) By $\text{Dyn}$’s definition, it is easy to prove that $\text{Reach}(H, v)[Z, Z, 0]$ holds for all $Z$. Hence if the formula $\text{Reach}(H, v)[Z, Z', T]$ holds, then $\text{Reach}(H, v)[Z, Z', T] \land \text{Reach}(H, v)[Z', Z', 0]$ holds too. It follows that there exist a $Z''$ and a $T'' \geq 0$ such that $\text{Reach}(H, v)[Z, Z'', T'] \land \text{Reach}(H, v)[Z'', Z', T - T']$, in particular, $Z'' = Z'$ and $T'' = T$.

($\Leftarrow$) Consider the formula

$$\phi[Z, Z', T] \overset{\text{def}}{=} \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, v)[Z, Z'', T'] \land$$

$$\text{Reach}(H, v)[Z'', Z', T - T'])$$

If there exist $p, q$ and $t, t'$ such that both $t = t'$ and $\phi[p, q, t]$ hold, then $\text{Reach}(H, v)[p, q, t] \land \text{Reach}(H, v)[q, q, 0]$, and thus $\phi[p, q, t]$ implies $\text{Reach}(H, v)[p, q, t]$. Moreover, if there exist $p, q$ and $t, t' \geq 0$ such that both $t' = 0$ and $\phi[p, q, t]$ hold, then $\text{Reach}(H, v)[p, p, 0] \land \text{Reach}(H, v)[p, q, t]$, indeed $\phi[p, q, t]$ implies $\text{Reach}(H, v)[p, q, t]$. Hence, in the following part of the proof, we consider the case in which both $T' \neq T$ and $T'' \neq 0$ hold. By $\text{Reach}$’s definition,

$$\text{Reach}(H, v)[Z, Z', T] \equiv ((T > 0 \land \text{Dyn}(v)[Z, Z', T] \land \text{Tp}(H, v)[Z, T]) \lor$$

$$(T = 0 \land Z = Z') \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'].$$
Hence the formula
\[ \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, v)[Z, Z'', T'] \land \text{Reach}(H, v)[Z'', Z', T - T']) \]
is equivalent to
\[ \exists Z'' \exists 0 \leq T' \leq T (((T' > 0 \land Tp(v)[Z, Z'', T']) \land Tp(H, v)[Z, T']) \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'']) \land \text{Dyn}(v)[Z'', Z', T - T'] \land \text{Tp}(H, v)[Z'', T - T'] \land ((T - T') = 0 \land Z'' = Z') \land \text{Inv}(v)[Z'] \land \text{Inv}(v)[Z'']) \]}

As said above, we are considering the case in which both \( T' \neq T \) and \( T' \neq 0 \) hold. In this case, it is easy to prove that the above formula is equivalent to
\[ \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, v)[Z, Z'', T'] \land \text{Reach}(H, v)[Z'', Z', T - T']) \]

Moreover, by \( Tp \)'s definition, if the formulæ \( Tp(H, v)[Z'', T - T'] \), \( Tp(H, v)[Z, T'] \), and \( \text{Reach}(H, v)[Z, Z'', T'] \) hold, then \( Tp(H, v)[Z, T] \) holds too. Hence, since \( Tp(H, v)[Z, T] \) is transitive, it follows that it is easy to prove that if the formula
\[ \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, v)[Z, Z'', T'] \land \text{Reach}(H, v)[Z'', Z', T - T']) \]
holds, then \( \text{Reach}(H, v)[Z, Z', T] \) holds too. \( \blacksquare \)

Notice that if \( ph \) is a path, then \( \text{Reach}(H, ph)[Z, Z', T] \) is a conjunction of formulæ of the kind \( \text{Reach}(H, v)[Z, Z', T] \) and \( \text{Reach}(H, e)[Z, Z'] \), where \( v \) is a location in \( ph \) and \( e \) is an edge. Hence, the above result can be extended to formulæ capturing reachability over a path.

**Lemma 4.3.5** Let \( H = (Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset}) \) be a hybrid automaton. Moreover, let \( ph = (v_i)_{i \in [0,M]} \) and \( ph' = (v'_i)_{i \in [0,M']} \) be two paths in \( (V,E) \) such that \( v_h = v'_h \). If \( Tp(H, v) \) is a transitive dynamics, then
\[ \text{Reach}(H, ph''[Z, Z', T]) \equiv \exists Z'' \exists 0 \leq T' \leq T (\text{Reach}(H, ph)[Z, Z'', T'] \land \text{Reach}(H, ph)[Z'', Z', T - T']) \]
where \( ph'' = ph \cdot ph' \).
Proof. Let $h''$ be the length of $ph \cdot ph'$ (i.e., $h'' = |ph \cdot ph'|$) and $ph \cdot ph'$ be the path $(\tau_i)_{i \in [0,h'']}$. By Reach’s definition,

$$\text{Reach}(H, ph'')[Z, Z', T] \equiv \exists Z^1, \ldots , Z^{2h''} \text{ Reach}(H, ph'')[Z, Z^1, \ldots , Z^{2h''}, Z', T].$$

Hence, by Reach’s definition,

$$\text{Reach}(H, ph'')[Z^0, Z^{2h''+1}, T] \equiv \exists Z^1, \ldots , Z^{2h''} \exists T_0 \geq 0 \ldots \exists T_{h''} \geq 0 \left( T = \sum_{i=0}^{h''} T_i \land \text{Reach}(H, \tau_0)[Z^0, Z^1, T_0] \land \bigwedge_{i \in [0,h''-1]} (\text{Reach}(H, (\tau_i, \tau_{i+1}))[Z^{2i+1}, Z^{2i+2}] \land \text{Reach}(H, \tau_{i+1})[Z^{2i+2}, Z^{2i+3}, T_{i+1}]) \right).$$

By Lemma 4.3.4, it follows that

$$\text{Reach}(H, ph'')[Z^0, Z^{2h''+1}, T] \equiv \exists Z^0 \exists 0 \leq T' \leq T \exists Z^1, \ldots , Z^{2h''} \exists T_0 \geq 0 \ldots \exists T_{h''} \geq 0 \left( T' = T + \sum_{i=0}^{h-1} T_i \land \text{Reach}(H, \tau_0)[Z^0, Z^1, T_0] \land \bigwedge_{i \in [0,h-2]} (\text{Reach}(H, (\tau_i, \tau_{i+1}))[Z^{2i+1}, Z^{2i+2}] \land \text{Reach}(H, \tau_{i+1})[Z^{2i+1}, Z^{2i+3}, T_{i+1}]) \land \text{Reach}(H, \tau_{h-1}, \tau_h)[Z^{2h-1}, Z^{2h}] \land \text{Reach}(H, \tau_h)[Z^{2h}, Z', T_h] \land \left( T - T' = T'' + \sum_{i=h+1}^{h''} T_i \land \text{Reach}(H, \tau_h)[Z'', Z^{2h+1}, T_h'] \land \bigwedge_{i \in [h,h'']-1} (\text{Reach}(H, \tau_i, \tau_{i+1})[Z^{2i+1}, Z^{2i+2}] \land \text{Reach}(H, \tau_{i+1})[Z^{2i+2}, Z^{2i+3}, T_{i+1}]) \right) \right).$$
\[
\equiv \exists Z'' \exists 0 \leq T' \leq T \left( \widehat{\text{Reach}}(H, ph)[Z, Z'', T'] \land \\
\widehat{\text{Reach}}(H, ph')[Z'', Z', T - T'] \right)
\]

Hence, the thesis holds.
4. Dynamics and Flow Selections


5

First-Order Constant Reset Hybrid Automata

“If I’d known computer science was going to be like this, I’d never have given up being a rock ‘n’ roll star.”

G. Hirst

In this chapter we introduce and study a special class of hybrid automata, First-Order Constant Reset hybrid automata (FOCoRe). Such automata are in Michael’s form and their resets are as in the class of O-minimal hybrid automata (see Section 3.4.2). Even though FOCoRe automata do not admit finite bisimulation quotient, we can translate reachability problems into satisfiability of a particular first-order formula over the reals. It follows that if the specifying theory is decidable, then the reachability problem is decidable.

5.1 FOCoRe Definition

A FOCoRe automaton is simply a hybrid automaton in Michael’s form whose resets are constant. More formally we can define it as follows.

Definition 5.1.1 (First-Order Constant Reset Automata) We say that a hybrid automaton $H$ is a first-order constant reset hybrid automaton, or simply a FOCoRe, if:

1. $H$ is in Michael’s form;
2. All the resets, $\text{Reset}(e)[Z, Z']$, of $H$ are constant i.e., they do not depend on $Z$.

Condition 1 will allow us to exploit Theorem 4.3.2 to check the existence of a valid continuous flows. Condition 2 is exactly the condition imposed on O-minimal hybrid automata (see Section 3.4.2).
Example 5.1.2 Let \( H = (Z, Z', V, \mathcal{E}, Inv, Dyn, Act, Reset) \) where:

- \( Z = (Z_1, Z_2) \) and \( Z' = (Z'_1, Z'_2) \);
- \( V = \{v\} \) and \( \mathcal{E} = \{e\} \), where \( e \) goes from \( v \) to \( v \);
- \( Inv(v)[Z] \equiv (0 \leq Z_1 \leq 1 \land 0 \leq Z_2 \leq 1) \);
- \( Dyn(v)[Z, Z', T] \equiv (Z'_1 = T + Z_1 \land Z'_2 = T^2 + Z_2) \);
- \( Act(e)[Z] \equiv (Z_1 = 1 \lor Z_2 = 1) \);
- \( Reset(e)[Z, Z'] \equiv (Z'_1 = 1 \land Z'_2 = 1) \).

The formulae in \( H \) are first-order formulæ over the reals. If \( p = (p_1, p_2) \), with \( 0 \leq p_1, p_2 \leq 1 \), then the function \( F^H_{v,p} \) is defined as \( F^H_{v,p}(t) = \{(q_1, q_2) | q_1 = t + p_1, q_2 \geq t^2 + p_2, \text{ and } 0 \leq q_1, q_2 \leq 1\} \). It is easy to see that \( p \in F^H_{v,p}(0) \) and for each \( t \) the set \( F^H_{v,p}(t) \) is closed and convex, since it is a segment. Moreover, this function is lower semi-continuous over the interval \( I^H_{v,p} \). It follows that \( H \) is in Michael’s form. Finally, \( Reset(e)[Z, Z'] \) does not depend on \( Z \). Hence, \( H \) is a FOCoRe automaton. ■

O-minimal hybrid automata are easily seen as special cases of FOCoRe automata. As a matter of fact, O-minimal hybrid automata allow only one continuous flow from each point, hence an O-minimal hybrid automaton is a FOCoRe for which the set \( F^H_{v,p}(t) \) reduces to a singleton, which is obviously closed and convex, for each time instant \( t \). The continuity of the flow immediately implies the lower semi-continuity of \( F^H_{v,p}(t) \) over \( I^H_{v,p} \). On the other hand, the class FOCoRe is not included in the class of O-minimal hybrid automata, since from each point we allow a set of flows. Moreover, FOCoReflows are not necessarily solutions of autonomous differential inclusions and their dynamics are not O-minimal in general.

Notice that the identification of a FOCoRe automaton can be checked automatically. In particular, in the remaining part of the section we present a first-order formula which holds if and only if the considered automaton is a FOCoRe.

As reported in Section 4.2, a hybrid automaton \( H \) is in Michael’s form if and only if the following formula holds:

\[
\bigwedge_{v \in V} \text{MForm}(H, v).
\]

Let us consider Condition 2 of FOCoRe definition. We just need to characterise the fact that, for all points \( p, p', q \in \mathbb{R}^k \), if \( Reset(e)[p, q] \) holds, then \( Reset(e)[p', q] \) does too. It is easy to prove that following formula expresses this fact.

\[
\text{ConstReset}(H, e) \overset{\text{def}}{=} \forall Z, Z^1, Z^2 \exists Z' \text{ Reset}(e)[Z, Z'] \rightarrow \text{ Reset}(e)[Z, Z']
\]
5.2 Reachability and Model Checking

Since both edges and locations are bounded, we can write the formula:

$$\bigwedge_{v \in V} \text{MForm}(H, e) \land \bigwedge_{e \in E} \text{ConstReset}(H, e)$$

which holds if and only if the corresponding hybrid automaton is a FOCoRe.

5.2 Reachability and Model Checking

Given a FOCoRe automaton $H$ and a starting region $R \subseteq \mathbb{R}^k$ characterised by a first-order formula $\rho$ over the reals, we may wish to compute the region $\text{ReachSet}(R) \subseteq \mathbb{R}^k$ of points that can be reached starting from a point in $R$ and following a trace of $H$.

More generally, given a formula $Q$ of a temporal logic, we may also be interested in determining the points of $R$ which satisfy $Q$. In the case of O-minimal hybrid automata, reachability as well as other temporal logic properties are checked through bisimulation as reported in Section 3.4.2. This technique can be applied whenever we consider a class $C$ of hybrid automata, which has the finite bisimulation property, i.e., each automaton in $C$ has a finite bisimulation quotient. Unfortunately, the class of FOCoRe does not possess the finite bisimulation property, as we will show in Section 5.3.

Our approach will instead exploit the properties of Michael’s form and constant resets. In particular, in this section, we demonstrate how the reachability problem over FOCoRe $\mathcal{T}$-automata can be reduced to the satisfiability of a first-order formula over the theory $\mathcal{T}$. From this it will follow the decidability of reachability problem over FOCoRe automata which are expressed in a decidable theory.

In Section 4.3, we showed the formula $\text{Reach}$ such that if $H$ is a hybrid automaton in Michael’s form, $\text{ph} = \langle v_0, \ldots, v_h \rangle$ is a path in $\langle V, E \rangle$ and $r, s \in \mathbb{R}^k$, then $r$ reaches $s$ in time $t$ through a trace $tr$ whose corresponding path is $\text{ph}$ if and only if the first-order formula $\text{Reach}(H, \text{ph})[r, Z^1, \ldots, Z^{2h}, s, t]$ is satisfiable. Unfortunately, as remarked in the same section, if $\langle V, E \rangle$ has a cycle, then it has an infinite number of paths and, thus the formula $\text{Reach}$ cannot be used to specify an effective method to reduce a reachability problem over $H$ to a satisfiability problem in a first-order theory. However, in the specific case of FOCoRe, we can exploit the fact that they have constant resets and ignore all the paths of $\langle V, E \rangle$ whose length exceeds $|E|$. Below, we denote the set of the path in $\langle V, E \rangle$ of length $|E|$ using the notation $\mathcal{P}_E$ and we write $\mathcal{P}_E(v)$ to denote the set of path in $\mathcal{P}_E$ stating in $v$.

**Theorem 5.2.1** Let $H = \langle Z, Z', V, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle$ be a FOCoRe automaton of dimension $k$. The point $s \in \mathbb{R}^k$ is reachable from $r \in \mathbb{R}^k$ by $H$ if and only if there exists a path $\text{ph} \in \mathcal{P}_E$ of length at most $|E|$ such that the formula $\exists T \geq 0 \text{Reach}(H, \text{ph})[r, s, T]$ holds.

**Proof.** ($\Leftarrow$) If there exists a path $\text{ph}$ such that $\exists T \geq 0 \text{Reach}(H, \text{ph})[r, s, T]$ holds, then, by definition of $\text{Reach}$, the formula $\exists T \geq 0 \text{Reach}(H, \text{ph})[r, Z^1, \ldots, Z^{2|\text{ph}|}, s, T]$ is satisfiable and, by Lemma 4.3.3, $r$ reaches $s$ in $H$. 
(⇒) Conversely, if \( s \in \text{ReachSet}(r) \), by Lemma 4.3.3, there exists a path \( ph \) such that the formula \( \exists T \geq 0 \text{Reach}(H, ph)[r, Z^1, \ldots, Z^{2|h|}, s, T] \) is satisfiable; let \( ph \) be the shortest of such paths. Thus, by definition of \( \text{Reach} \), the formula \( \exists T \geq 0 \text{Reach}(H, ph)[r, s, T] \) holds. Moreover, if the length of \( ph \) is less than or equal to \( |E| \), then \( ph \in \overline{P}_E \) and we are done. If, on the other hand, \( ph \) is greater than \( |E| \), then \( ph \) is of the form \( \langle v_0, v_1, \ldots, v_h \rangle \) with \( h > |E| \). Hence, by the pigeonhole principle, there must exist at least one repeated subsequence \( v_i, v_{i+1} \) in \( ph \). Let \( ph' \) be the path obtained from \( ph \) by removing all such repetitions, i.e.: if in \( ph \) there is a subsequence of the form \( v_i, v_{i+1}, \ldots, v_j, v_{j+1}, v_{j+2}, \) with \( v_i = v_j \) and \( v_{i+1} = v_{j+1} \), then we replace it with \( v_i, v_{i+1}, v_{j+2} \). Since we can show that \( ph' \) satisfies all the requirements and since it is strictly shorter than \( ph \), we derive a contradiction. In the following, we prove that \( \exists T \geq 0 \text{Reach}(H, ph')[r, \ldots, s, T] \) is satisfiable. It is sufficient to prove the thesis in the case \( ph' \) has been obtained from \( ph \) with only one removal. Let \( ph \) be of the form \( v_0, \ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, v_{j+2}, \ldots, v_h \) with \( v_i = v_j \) and \( v_{i+1} = v_{j+1} \) and \( ph' \) be \( v_0, \ldots, v_i, v_{i+1}, v_{j+2}, v_{2h} \). The formula \( \text{Reach}(H, ph)[r, \ldots, s, T] \) is of the form:

\[
\exists T_0 \geq 0 \ldots \exists T_h \geq 0 \left( T = \sum_{l=0}^{h} T_l \wedge \text{Reach}(H, v_0)[r, Z^1, T_0] \wedge \ldots \wedge \text{Reach}(H, v_i)[Z^{2i}, Z^{2i+1}, T_i] \wedge \text{Reach}(H, \langle v_i, v_{i+1} \rangle)[Z^{2i+1}, Z^{2(i+1)}] \wedge \ldots \wedge \text{Reach}(H, v_j)[Z^{2j}, Z^{2j+1}, T_j] \wedge \text{Reach}(H, \langle v_j, v_{j+1} \rangle)[Z^{2j+1}, Z^{2(j+1)}] \wedge \ldots \wedge \text{Reach}(H, v_{j+1}, v_{j+2})][Z^{2(j+1)+1}, Z^{2(j+2)}] \wedge \ldots \wedge \text{Reach}(H, v_h)[Z^{2h}, s, T_h] \right),
\]

while the formula \( \text{Reach}(H, ph')[r, \ldots, s, T] \) is of the form:

\[
\exists T_0 \geq 0 \ldots \exists T_{i+1} \geq 0 \left( T = \sum_{l=0}^{i+1} T_l + \sum_{l=i+2}^{h} T_l \right) \wedge \text{Reach}(H, v_0)[r, Z^1, T_0] \wedge \ldots \wedge \text{Reach}(H, v_i)[Z^{2i}, Z^{2i+1}, T_i] \wedge
\]
where we keep the indexing of \( ph \) from \( j + 2 \) to \( 2h \).

Let us assume that \( \exists T \geq 0 \) \( \mathsf{Reach}(H, ph)[r, \ldots, s, T] \) can be satisfied by replacing \( Z^a \) with \( z^a \) for each \( a \leq 2h \). In order to satisfy \( \exists T \geq 0 \mathsf{Reach}(H, ph')[r, \ldots, s, T] \) we replace \( Z^a \) by \( z^a \) for each \( a \neq 2(i + 1), 2(i + 1) + 1 \). Moreover, we replace \( Z^{2(i+1)} \) by \( z^{2(j+1)} \) and \( Z^{2(i+1)+1} \) by \( z^{2(j+1)+1} \). In the following part of the proof, we prove that such a replacement satisfies \( \exists T \geq 0 \mathsf{Reach}(H, ph')[r, \ldots, s, T] \). Since the first replacement satisfies \( \exists T \geq 0 \mathsf{Reach}(H, ph)[r, \ldots, s, T] \), we have that both the formulæ

\[
\mathsf{Inv}(v_i)[z^{2i+1}] \land \mathsf{Act}((v_i, v_{i+1}))[z^{2i+1}] \land \\
\mathsf{Reset}((v_i, v_{i+1}))[z^{2i+1}] \land \mathsf{Inv}(v_{i+1})[z^{2i+1}]
\]

and

\[
\mathsf{Inv}(v_i)[z^{2j+1}] \land \mathsf{Act}((v_i, v_{i+1}))[z^{2j+1}] \land \\
\mathsf{Reset}((v_i, v_{i+1}))[z^{2j+1}] \land \mathsf{Inv}(v_{i+1})[z^{2j+1}]
\]

hold. It follows that \( \mathsf{Inv}(v_i)[z^{2i+1}] \land \mathsf{Act}((v_i, v_{i+1}))[z^{2i+1}] \land \mathsf{Reset}((v_i, v_{i+1}))[z^{2(j+1)}] \land \mathsf{Inv}(v_{i+1})[z^{2(j+1)}] \) also holds, thus \( \mathsf{Reach}(H, (v_i, v_{i+1}))[z^{2i+1}, z^{2(j+1)}] \) is true. The rest of the proof follows from the fact that the replacement satisfies the formula \( \exists T \geq 0 \mathsf{Reach}(H, ph)[r, \ldots, s, T] \). Hence \( \exists T \geq 0 \mathsf{Reach}(H, ph')[r, \ldots, s, T] \) is satisfiable, and the formula \( \exists T \geq 0 \overline{\mathsf{Reach}(H, ph')[r, s, T]} \) holds, by definition of \( \overline{\mathsf{Reach}} \).

Given a FOCoRe automaton \( H = (Z, Z', V, E, \mathsf{Inv}, \mathsf{Dyn}, \mathsf{Act}, \mathsf{Reset}) \), if \( \mathcal{T}_E \) is the set of paths of \( (V, E) \) of length at most \( |E| \), we can define the first-order formula \( \mathcal{P}_H[Z, Z'] \) as follows:

\[
\mathcal{P}_H[Z, Z'] \overset{\text{def}}{=} \bigvee_{ph \in \mathcal{T}_E} \exists T \geq 0 \overline{\mathsf{Reach}(H, ph)[Z, Z', T]}.
\]

From Theorem 5.2.1, it follows that, given a FOCoRe \( H, s \in \mathsf{ReachSet}_H(r) \) if and only if the formula \( \mathcal{P}_H[r, s] \) holds. We can now characterise the set of points reachable from a first-order definable set \( R \subseteq \mathbb{R}^k \).

**Corollary 5.2.2** Let \( H \) be a FOCoRe automaton and \( \rho[Z] \) be a first-order formula. The set \( \mathsf{ReachSet}_H(\mathsf{Sat}(\rho)) \) is characterised by the first-order formula

\[
S_H(\rho)[Z'] \overset{\text{def}}{=} \exists Z (\rho[Z] \land \mathcal{P}_H[Z, Z']).
\]
Thus we have reduced our reachability problem to that of deciding the satisfiability of an existential first-order formula and, hence, we get the following corollary.

**Corollary 5.2.3** Let $T$ be a decidable theory and $H$ be a FOCoRe. If $H$ is a $T$-automaton, then the reachability problem for $H$ is decidable.

### 5.3 FOCoRe and Bisimulation

In this section we prove that there exists a FOCoRe which does not admit a finite bisimulation quotient. In particular, we prove that the hybrid automaton $H_{inf} = (Z, Z', V, E, Inv, Dyn, Act, Reset)$ where:

- $Z = (Z_1, Z_2)$ and $Z' = (Z'_1, Z'_2)$, where $Z_1, Z_2, Z'_1$ and $Z'_2$ are variables over $\mathbb{R}$,
- $V = \{v\}$ and $E = \{e\}$, where $e$ goes form $v$ to $v$,
- $Inv(v)[Z] \equiv (-1 \leq Z_1 \leq 1 \land Z_2 > 0)$,
- $Dyn(v)[Z, Z', t] \equiv up[Z, Z'] \land up'[Z, Z'] \land \|Z' - Z\| \leq T$, where $up[Z, Z'] \equiv Z'_2 \geq Z_2Z'_1 + Z_2(1 - Z_1)$ and $up'[Z, Z'] \equiv Z'_2 \geq -Z_2Z'_1 + Z_2(1 + Z_1)$,
- $Act(e)[Z] \equiv (Z_1 = 1 \land 0 < Z_2 \leq 1)$,
- $Reset(e)[Z, Z'] \equiv (Z'_1 = -1 \land 0 < Z'_2 \leq 1)$,

is a FOCoRe and does not admit a finite bisimulation quotient.

**Lemma 5.3.1** $H_{inf}$ is a FOCoRe automaton.
Proof. To prove that $H_{\text{inf}}$ is a FOCoRe automaton, we need to show that it is in Michael's form and that its resets are constant. Condition 2 of FOCoRe definition and Condition 1 of Michael's form definition are obviously satisfied. To prove Condition 2 of Michael's form definition, we have to prove that for each $v \in V$ and $p = (p_1, p_2) \in \mathbb{R}^2$ such that $\text{Inv}(v)[p]$ holds, the function $F^H_{v,p}$ is lower semi-continuous and, for all $t \in I^H_{v,p}$, $F^H_{v,p}(t)$ is a closed and convex set. As we have defined $H_{\text{inf}}$, for all $t \in \mathbb{R}_{\geq 0}$, $\text{Dyn}(v)[p, Z', t] \equiv Z'_2 \geq p_2 Z'_1 + p_2 (1 - p_1) \land Z'_2 \geq -p_2 Z'_1 + p_2 (1 + p_1) \land \|Z' - p\| \leq t$, where $Z' = (Z'_1, Z'_2)$. Thus for all $t \in \mathbb{R}_{\geq 0}$ and all $p \in \mathbb{R}^2$, $\text{Dyn}(v)[p, p, t]$ holds and, if $\text{Inv}(v)[p]$ holds, for all $t \in \mathbb{R}_{\geq 0}$, $\text{Dyn}(v)[p, p, t] \land \text{Inv}(v)[p]$ holds too. Hence, for all $t \in \mathbb{R}_{\geq 0}$, the formula $\exists Z' \ (\text{Dyn}(v)[p, Z', t] \land \text{Inv}(v)[Z'])$ is true. It follows that $F^H_{v,p} = [0, +\infty)$. We now prove that $F^H_{v,p}$ is convex. For all $t \in I^H_{v,p}$, $F^H_{v,p}$ is such that $F^H_{v,p}(t) = \{ q \mid \text{Dyn}(v)[p, q, t] \land \text{Inv}(v)[q]\}$, where $q = (q_1, q_2)$. Hence, by $\text{Dyn}$'s definition, $F^H_{v,p}(t) = \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v)) \cap \text{Sat}(\|p - Z\| \leq t)$. Since the intersection of convex sets is convex, to deduce the convexity of $F^H_{v,p}(t)$, we will prove the convexity of $\text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v))$. A set $S$ is convex if and only if for all $q, q \in S$, all points of the segment between $q$ and $q$ are contained in $S$. The convexity of $\text{Sat}(\|p - Z\| \leq t)$ is obvious, hence we have to prove the convexity of $\text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v))$. In particular, we need to prove that for all $p = (p_1, p_2)$, $q = (q_1, q_2)$, $r = (r_1, r_2) \in \mathbb{R}^2$, and for all $\alpha \in [0,1]$, if $q, r \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v))$ then $(s_1, s_2) \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v))$, where $s_1 = (1 - \alpha)q_1 + \alpha r_1$ and $s_2 = (1 - \alpha)q_2 + \alpha r_2$. If $q \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z])$ then $q_2 \geq p_2 q_1 + p_2 (1 - p_1) \land q_2 \geq -p_2 q_1 + p_2 (1 + p_1)$ and if $r \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z])$ then $r_2 \geq p_2 r_1 + p_2 (1 - p_1) \land r_2 \geq -p_2 r_1 + p_2 (1 + p_1)$. Thus:

\[
\begin{align*}
s_2 &= (1 - \alpha)q_2 + \alpha r_2 \\
&\geq (1 - \alpha)(p_2 q_1 + p_2 (1 - p_1)) + \alpha (p_2 r_1 + p_2 (1 - p_1)) \\
&\geq p_2 ((1 - \alpha)q_1 + \alpha r_1) + p_2 (1 - p_1) ((1 - \alpha) + \alpha).
\end{align*}
\]

But, $s_1 = (1 - \alpha)q_1 + \alpha r_1$ hence:

\[
\begin{align*}
s_2 \geq p_2 ((1 - \alpha)q_1 + \alpha r_1) + p_2 (1 - p_1) ((1 - \alpha) + \alpha) \\
&\geq p_2 ((1 - \alpha)q_1 + \alpha r_1) + p_2 (1 - p_1) \\
&\geq p_2 s_1 + p_2 (1 - p_1).
\end{align*}
\]

Symmetrically:

\[
\begin{align*}
s_2 &= (1 - \alpha)q_2 + \alpha r_2 \\
&\geq (1 - \alpha)(p_2 (1 + p_1) - p_2 q_1) + \alpha (p_2 (1 + p_1) - p_2 q_1) \\
&\geq -p_2 ((1 - \alpha)q_1 + \alpha r_1) + p_2 (1 + p_1) ((1 - \alpha) + \alpha) \\
&\geq -p_2 s_1 + p_2 (1 + p_1),
\end{align*}
\]

thus, for all $s$ laying on the segment between $q$ and $r$, the formula $\text{up}[p, s] \land \text{up}'[p, s]$ holds. Moreover, if $\text{Inv}(v)[q]$ and $\text{Inv}(v)[r]$ then $-1 \leq q_1 \leq 1 \land q_2 > 0$ and $-1 \leq
5. First-Order Constant Reset Hybrid Automata

\[ r_1 \leq 1 \land r_2 > 0, \text{ thus } s_2 = (1 - \alpha)q_2 + \alpha r_2 \geq (1 - \alpha)0 + \alpha 0 \geq 0. \]

Furthermore, \( s_1 = (1 - \alpha)q_1 + \alpha r_1 \geq -(1 - \alpha) - \alpha \geq -1 \) and \( s_1 = (1 - \alpha)q_1 + \alpha r_1 \leq (1 - \alpha) + \alpha \leq 1 \) and hence, for all \( s \) belonging to the segment between \( q \) and \( r \), the formula \( \text{Inv}(v)[s] \) holds. Thus for all \( q, r \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v)) \) and for all \( s \) belonging to the segment between \( q \) and \( r \), \( s \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z]) \cap \text{Sat}(\text{Inv}(v)) \). Hence we proved the convexity of \( F_{v,p}^H(t) \).

We now prove that \( F_{v,p}^H \) is lower semi-continuous. By Definition 4.1.1, \( F_{v,p}^H \) is lower semi-continuous if and only if for all \( q \in F_{v,p}^H(t) \) and for all neighbourhoods \( U_{q,\epsilon} = \{q' ||q' - q| < \epsilon\} \) of \( q \) there exists a neighbourhood \( U_{t,\delta} = \{t' ||t' - t| < \delta\} \) of \( t \) such that \( \forall t' \in U_{t,\delta} \) the set \( (F_{v,p}^H(t') \cap U_{q,\epsilon}) \) is not empty. Now we prove that, for all \( q \in F_{v,p}^H(t) \) and for all \( \epsilon > 0 \), \( \delta = \frac{\epsilon}{2} \) is such that \( \forall t' \in U_{t,\delta} \) \( F_{v,p}^H(t') \cap U_{q,\epsilon} \neq \emptyset \). By constant reset condition in the FOCoRe's definition. In particular, \( \epsilon > 0 \) then \( F_{v,p}^H(t') \cap U_{q,\epsilon} \neq \emptyset \). So assume that \( t' < t \). By definition of \( v \), it follows directly that \( ||r - q|| + ||p - r|| = ||p - q|| \). Moreover, since \( \epsilon > 0 \) then \( ||p - r|| \leq t' - \frac{\epsilon}{2} \). Hence \( \epsilon > 0 \) then \( ||p - r|| \leq t' \).

Moreover, since \( r \in F_{v,p}^H(t) \), the formula \( \text{up}[p, r] \land \text{up}'[p, r] \land \text{Inv}(v)[r] \) holds. Hence \( r \in \text{Sat}(\text{up}[p, Z] \land \text{up}'[p, Z] \land \text{Inv}(v)) \cap \text{Sat}([p - Z| \leq t']) \). By our automaton, if \( p = (p_1, p_2) \) and \( q = (q_1, q_2) \), then \( \text{up}[p,q] \land \text{up}'[p,q] \equiv q_2 \geq p_2 q_1 + p_2 (1 - p_1) \land q_2 \geq -p_2 q_1 + p_2 (1 + p_1) \land ||p - q|| \leq t \). Moreover, the formula \( q_2 \geq -p_2 q_1 + p_2 (1 + p_1) \) holds if and only if \( p_2 q_1 \geq -q_2 + p_2 (1 + p_1) \) is true. Thus, from \( \text{Dyn}(v)[p, q, t] \), it follows that:

\[
q_2 \geq p_2 q_1 + p_2 (1 - p_1) \\
\geq -q_2 + p_2 (1 + p_1) + p_2 (1 - p_1) \\
\geq -q_2 + 2 p_2,
\]

and hence \( q_2 \geq p_2 \). Since \( \text{Inv}(v)[q] \equiv -1 \leq q_1 - 1 \land q_2 > 0 \) and \( q_2 > p_2 \land \text{Inv}(v)[p] \) implies \( q_2 > 0 \), for all \( p \in \mathbb{R}^2 \) such that \( \text{Inv}(v)[p] \) and for all \( t \in I_{v,p}^H \), \( F_{v,p}^H(t) = \{q | q_2 \geq p_2 q_1 + p_2 (1 - p_1) \land q_2 \geq -p_2 q_1 + p_2 (1 + p_1) \land -1 \leq q_1 \leq 1 \land ||p - q|| \leq t\} \), where \( q = (q_1, q_2) \). Hence, since \( F_{v,p}^H(t) \) is an intersection of closed sets, \( F_{v,p}^H(t) \) is a closed set. It follows that \( H_{\text{inf}} \) is a FOCoRe automaton.

To prove that the automaton \( H_{\text{inf}} \) does not admit finite bisimulation quotient, we have to exploit the constant reset condition in the FOCoRe’s definition. In particular, by \( \text{Pre}_{\sigma}(P) \)’s definition, and by constant reset condition, it follows that:

\[
\text{Pre}_{\epsilon}(P) = \begin{cases} 
\emptyset & \text{if } P \cap \mathcal{R}(e) = \emptyset \\
A(e) & \text{if } P \cap \mathcal{R}(e) \neq \emptyset
\end{cases}
\]
Thus, as reported in [90], $H_{\inf}$ admits a finite bisimulation quotient if and only if Algorithm 2 terminates, when the initial partition is the partition $S_0$ induced by the set $A_0 = \{I(v)\} \cup \bigcup_{(v',v) \in E} \{R((v',v))\} \cup \bigcup_{(v,v') \in E} \{A((v,v'))\}$.

**Algorithm 2** Bisimulation algorithm for hybrid systems with constant resets

```
for v ∈ V do
    S_v ← compute_initial_partition_from(A_v)
    while ∃P, P' ∈ S_v such that ∅ ≠ P ∩ Pre_v (P') ≠ P do
        P_1 ← P ∩ Pre_v (P')
        P_2 ← P \ Pre_v (P')
        S_v ← (S_v \ {P}) ∪ {P_1, P_2}
    end while
end for
X/ ∼= \bigcup_v \langle v, S_v \rangle
```

However, following results allow us to conclude that Algorithm 2 does not terminate on $H_{\inf}$ and consequently, $H_{\inf}$ does not admit finite bisimulation quotient. Below, we prove that, considering the $H_{\inf}$ automaton, there exists two sets satisfying the while condition at the end of each cycle of Algorithm 2. In particular, Lemma 5.3.3 and Lemma 5.3.4 prove that each algorithm iteration adds to $S_0$ a non-empty set $P$ smaller than $P'$ such that $P'$ satisfies the while condition.

**Lemma 5.3.2** For the automaton $H_{\inf}$, if the formula $Inv(v)[p]$ holds then, for each $t ∈ \mathbb{R}_{\geq 0}$, $Tp(H,v)[p,t]$ is true.

**Proof.** By definition, $Tp(H,v)[p,t] ≜ ∀0 ≤ T' ≤ t \exists Z' Dyn(v)[p, Z', T'] ∧ Inv(v)[Z']$. Moreover, by $H_{\inf}$’s definition, $Dyn(v)[p, Z', T] ≜ Z'_p ≥ p_2 Z'_1 + p_2 1 - p_1 \land Z'_2 ≥ -p_2 Z'_1 + p_2 (1 + p_1) \land \|p - Z'\| ≤ T$ and $Inv(v)[p] ≜ -1 ≤ p_1 ≤ 1 \land p_2 > 0$, where $p = (p_1, p_2)$ and $Z' = (Z'_1, Z'_2)$. It follows that, for all $t ∈ \mathbb{R}_{\geq 0}$, $Dyn(v)[p, p, t]$ holds. Thus if $Inv(v)[p]$ holds then, for all $t \in \mathbb{R}_{\geq 0}$, $Z'\ Dow(v)[p, Z', t] ∧ Inv(v)[Z']$ is true. Hence, by definition of the formula $Tp(H,v)[p,t]$, if $Inv(v)[p]$ holds then, for all $t \in \mathbb{R}_{\geq 0}$, $Tp(H,v)[p,t]$ holds too. □

**Lemma 5.3.3** Let $G(r)$ be the subset of $\mathbb{R}^2$ such that $G(r) ≜ \{(p_1, p_2) | p_1 = 1 \land 0 < p_2 ≤ r\}$. For the automaton $H_{\inf}$, $Pre_v (G(r)) = \{p | 3p_2 ≤ r(p_1 + 2) ∧ Inv(v)[p]\}$, where $p = (p_1, p_2)$ and $v ∈ V$.

**Proof.** By definitions, $G(r) = \{(p_1, p_2) | p_1 = 1 \land 0 < p_2 ≤ r\}$ and $Inv(v)[(p_1, p_2)] ≜ p_2 > 0 \land -1 ≤ p_1 ≤ 1$. Hence, each point $p$ in $G(r)$ is such that $Inv(v)[p]$ and then, by Lemma 5.3.2 for each $t ∈ \mathbb{R}_{\geq 0}$ it holds that $Tp(H,v)[p,t]$. Thus, from Theorem 4.3.2, it follows that $Pre_v (G(r)) = \{p ∈ \mathbb{R}^2 | ∃q ∈ G(r) ∃T ≥ 0 Dyn(v)[p,q,T] ∧ Inv(v)[p]\}$. We can now prove that for all $(p_1, p_2) ∈ I(v)$ the formula $∃q ∈ G(r) ∃T ≥ 0 Dyn(v)[(p_1, p_2), q, T]$ holds if and only if $p_2 ≤ \frac{1}{3}(p_1 + 2)$ holds. We proceed as follows: first we show that, for all $(p_1, p_2) ∈ I(v)$, if $p_2 ≤ \frac{1}{3}(p_1 + 2)$ does not hold then neither does $∃q ∈ G(r) ∃T ≥ 0 Dyn(v)[(p_1, p_2), q, T]$ (point 1); next we show
that, for all \((p_1, p_2) \in \mathcal{I}(v)\), \(\neg(\exists q \in G(r) \exists T \geq 0 \, \text{Dyn}(v)[(p_1, p_2), q, T])\) implies the formula \(p_2 > \frac{r}{3}(p_1 + 2)\) (point 2).

1. By definition, \(\text{Dyn}(v)[p, q, T] \equiv q_2 \geq p_2q_1 + p_2(1 - p_1) \land q_2 \geq -p_2q_1 + p_2(1 + p_1) \land \|p - q\| \leq T\). Thus, if we assume by contradiction that both conditions, \(p_2 > \frac{r}{3}(p_1 + 2)\) and \(\exists q \in G(r) \exists T \geq 0 \, \text{Dyn}(v)[(p_1, p_2), q, T]\), hold then:

\[
q_2 \geq p_2q_1 + p_2(1 - p_1) \\
> \frac{r}{3}(p_1 + 2)(q_1 + (1 - p_1))
\]

Since \((q_1, q_2) \in G(r)\) and \((p_1, p_2) \in \mathcal{I}(v)\), it follows that \(q_1 = 1\) and \(p_1 \leq 1\) hence:

\[
q_2 > \frac{r}{3}(p_1 + 2)(q_1 + (1 - p_1)) \\
> \frac{r}{3}(p_1 + 2)(2 - p_1) \\
> \frac{r}{3}(4 - p_1^2) > \frac{r}{3}(4 - 1) \\
> r.
\]

But, by definition, \(G(r) = \{(q_1, q_2) \mid q_1 = 1 \land 0 < q_2 \leq r\}\). Hence, the equation above contradicts our initial hypothesis. Thus, for all \((p_1, p_2) \in \mathcal{I}(v)\), if \(p_2 > \frac{r}{3}(p_1 + 2)\) holds then \(\exists q \in G(r) \exists T \geq 0 \, \text{Dyn}(v)[(p_1, p_2), q, T]\) does not.

2. By definition, \(\text{Dyn}(v)[p, q, T] \equiv q_2 \geq p_2q_1 + p_2(1 - p_1) \land q_2 \geq -p_2q_1 + p_2(1 + p_1) \land \|p - q\| \leq T\), thus if we assume by contradiction that both formulæ \(\forall q \in G(r) \forall T \geq 0 \, \neg\text{Dyn}(v)[p, q, T]\) and \(p_2 \leq \frac{r}{3}(p_1 + 2)\) hold then either \(q_2 < p_2q_1 + p_2(1 - p_1)\), \(q_2 < -p_2q_1 + p_2(1 + p_1)\) or \(\forall q \in G(r) \forall T \geq 0 \|p - q\| > T\). If the formula \(q_2 < p_2q_1 + p_2(1 - p_1)\) holds then:

\[
q_2 < p_2q_1 + p_2(1 - p_1) \\
< \frac{r}{3}(p_1 + 2)(q_1 + (1 - p_1))
\]

Since \((q_1, q_2) \in G(r)\) and \((p_1, p_2) \in \mathcal{I}(v)\), it follows that \(q_1 = 1\) and \(p_1 \geq -1\) hence:

\[
q_2 < \frac{r}{3}(p_1 + 2)(q_1 + (1 - p_1)) \\
< \frac{r}{3}(p_1 + 2)(2 - p_1) \\
< \frac{r}{3}(4 - p_1^2) < \frac{r}{3}(4 - 1) \\
< r.
\]

But, by definition, \(G(r) = \{(q_1, q_2) \mid q_1 = 1 \land 0 < q_2 \leq r\}\) and, in particular, \((1, r) \in G(r)\). Hence, the formula \(q_2 < p_2q_1 + p_2(1 - p_1)\) contradicts our hypothesis.
Let assume that the formula \( q_2 < -p_2q_1 + p_2(1 + p_1) \) holds. Since \((q_1, q_2) \in G(r)\) and \((p_1, p_2) \in I(v)\), by hypothesis, \( q_1 = 1 \) and \( p_1 \leq 1 \). It follows that
\[
q_2 < -p_2q_1 + p_2(1 + p_1)
\]
\[
< -p_2 + p_2(1 + 1) = p_2.
\]
Moreover, the formula \( p_2 \leq \frac{r}{3}(p_1 + 2) \) holds by hypothesis, thus
\[
q_2 < p_2 \leq \frac{r}{3}(p_1 + 2) \leq \frac{r}{3} = r
\]
But by definition, \( G(r) = \{ (q_1, q_2) \mid q_1 = 1 \land 0 < q_2 \leq r \} \) and, in particular, \((1, r) \in G(r)\). Hence, the formula \( q_2 < -p_2q_1 + p_2(1 + p_1) \) contradicts our hypothesis.

Let assume that \( \forall q \in G(r) \forall T \geq 0 ||p - q|| > T \) holds. Let us over-estimate the maximum of \( ||p - q|| \) when \( p \in I(v) \), \( q \in G(r) \), and \( p_2 \leq \frac{r}{3}(p_1 + 2) \).

\[
\max ||p - q|| \leq \max \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}
\]
\[
\leq \sqrt{\max (p_1 - q_1)^2 + \max (p_2 - q_2)^2}
\]
\[
\leq \sqrt{\max (\max p_1 - \min q_1, \min p_1 - \max q_1)^2 + \max (p_2 - q_2)^2}
\]
Since \((q_1, q_2) \in G(r)\), \((p_1, p_2) \in I(v)\), and \( p_2 \leq \frac{r}{3}(p_1 + 2) \) by hypothesis, it follows that \( q_1 = 1 \), \( q_2 \in (0, r] \), \( p_1 \in [-1, 1] \), and \( p_2 > 0 \). Moreover:
\[
p_2 \leq \frac{r}{3} (p_1 + 2)
\]
\[
\leq \frac{r}{3} (1 + 2) = r
\]
Thus:
\[
\max ||p - q|| \leq \sqrt{\max (\max p_1 - \min q_1, \min p_1 - \max q_1)^2 + \max (p_2 - q_2)^2}
\]
\[
\leq \sqrt{\max (1 - 1, -1 - 1)^2 + \max (p_2 - q_2)^2}
\]
\[
\leq \sqrt{4 + \max (\max p_2 - \min q_2, \min p_2 - \max q_2)^2}
\]
\[
\leq \sqrt{4 + \max (r - 0, 0 - r)^2}
\]
\[
\leq \sqrt{4 + r^2}
\]
It follows that \( \sqrt{4 + r^2} \) is greater or equal to ||\(p - q\)|| for all \( q \in G(r)\) and all \( p \in I(v)\) satisfying \( p_2 \leq \frac{r}{3}(p_1 + 2) \). Hence, the formula \( \forall q \in G(r) \forall T \geq 0 ||p - q|| > T \) contradicts our hypothesis.

Thus, for all \((p_1, p_2) \in I(v)\), if \( \lnot (\exists q \in G(r) \exists T \geq 0 \Dyn(v)([p_1, p_2], q, T)) \) holds then so does \( p_2 > \frac{r}{3}(p_1 + 2) \).
It follows that $\text{Pre}_v(G(r)) = \{(p_1, p_2) \mid p_2 \leq \frac{r}{3}(p_1 + 2) \wedge \text{Inv}(v)[p]\}$.  

Lemma 5.3.4  Let $L(r)$ be the subset of $\mathbb{R}^2$ such that $L(r) \overset{\text{def}}{=} \{(p_1, p_2) \mid p_1 = -1 \wedge 0 < p_2 \leq r\}$. The automaton $H_{\inf}$ satisfies $\text{Pre}_v(L(r)) = \{p \mid 3p_2 \leq r(2 - p_1) \wedge \text{Inv}(v)[p]\}$, where $p = (p_1, p_2)$ and $v \in V$.

Proof. The proof is analogous to the proof of Lemma 5.3.3.

Notice that, for the automaton $H_{\inf}$, $L(1)$ and $G(1)$ are equal to $\mathcal{R}(e)$ and $\mathcal{A}(e)$, respectively.

Theorem 5.3.5  The automaton $H_{\inf}$ does not admit finite bisimulation quotient.

Proof. Our proof that $H_{\inf}$ does not admit finite bisimulation quotient relies on showing that Algorithm 2 does not terminate on $H_{\inf}$. At the start of the computation, Algorithm 2 uses $S_v = (\mathcal{R}(e), \mathcal{A}(e), \mathcal{I}(v) \setminus (\mathcal{R}(e) \cup \mathcal{A}(e)))$ as an initial partition. As $L(1) = \mathcal{R}(e)$ and $G(1) = \mathcal{A}(e)$, $S_v = \{L(1), G(1), \mathcal{I}(v) \setminus (L(1) \cup G(1))\}$. If $p = (p_1, p_2)$ then, by Lemma 5.3.4 and $G$’s definition:

$$\text{Pre}_v(L(r)) \cap G(r') = \{Z \mid p_2 \leq \frac{r}{3}(2 - p_1) \wedge \text{Inv}(v)[Z] \wedge p_1 = 1 \wedge 0 < p_2 \leq r'\}$$

$$= \{Z \mid p_2 \leq \frac{r}{3} \wedge \text{Inv}(v)[Z] \wedge p_1 = 1 \wedge 0 < p_2 \leq r'\}$$

$$= G\left(\frac{r}{3}\right).$$

Similarly, by Lemma 5.3.3 and $L$’s definition: $\text{Pre}_v(G(r')) \cap L(r) = L\left(\frac{r'}{3}\right)$. Thus, if $r < 3r'$ and $r, r' \in \mathbb{R}_{\geq 0}$ then $\emptyset \neq \text{Pre}_v(L(r)) \cap G(r') \neq G(r')$ and then the
algorithm removes $G(r')$ from $S_v$ and it inserts the sets $G\left(\frac{r}{3}\right)$ and $G(r') \setminus G\left(\frac{r}{3}\right)$ in $S_v$. Otherwise, $r \geq 3r'$ holds and if $r, r' \in \mathbb{R}_{\geq 0}$ then $3r > r \geq 3r' > r'$. It follows that $\emptyset \neq \text{Pre}_v(G(r')) \cap L(r) \neq L(r)$ and then the algorithm removes $L(r)$ from $S_v$ and it inserts the sets $L\left(\frac{r}{3}\right)$ and $L(r) \setminus L\left(\frac{r}{3}\right)$ in $S_v$. Hence, since the initial partition contains both $L(1)$ and $G(1)$, during the subsequent computation steps, there will exist $r, r' \in (0, 1]$ such that $L(r), G(r') \in S_v$. Moreover, at each computation steps $\exists P, P' \in S_v \mid \emptyset \neq \text{Pre}_v(P) \cap P' \neq P'$ in particular, if $r < 3r'$ then $P = L(r)$ and $P' = G(r')$, since, Otherwise, $P = G(r')$ and $P' = L(r)$. It follows then that Algorithm 2 does not terminate, leading to the conclusion that $H_{\inf}$ does not admit finite bisimulation.

Hence, the next corollary follows from Lemma 5.3.1 and Theorem 5.3.5.

**Corollary 5.3.6** There exist FOCoRe automata that do not admit finite bisimulation quotient.

**Proof.** By Lemma 5.3.1, $H_{\inf}$ is a FOCoRe automaton and, by Theorem 5.3.5, $H_{\inf}$ does not admit finite bisimulation quotient.

**Remark 5.3.7** Notice that the finiteness of bisimulation quotient does not depend on the decidability of the theory used to define the FOCoRe. As a matter of fact, $H_{\inf}$ is a semi-algebraic hybrid automaton and, as remarked in Chapter 1, the theory used to defined it is decidable.

## 5.4 Model Checking

Despite the absence of a finite bisimulation result for FOCoRe, building upon the decidability of the reachability problem, we can still show that a substantial and interesting fragment of $\text{CTL}_{\rightarrow}$ can be decided over FOCoRe automata. Notice that
since this fragment is not included in LTL it is not possible to use simulation equivalence to reduce the model.

Given a FOC
CoRe automaton $H$ of dimension $k$, we consider a set $\mathcal{P} = \{P_1[Z], \ldots, P_m[Z]\}$ of atomic propositions whose elements are first-order formulæ over the reals with $k$ free-variables. Let $\Phi_P$ be the set of formulæ defined by the following grammar.

$$Q ::= P[Z] \mid \neg P[Z] \mid Q_1 \lor Q_2 \mid E_3 \mid A_2 Q_1$$

Consider the two simulation equivalent models in Figure 2.3(a), where $A$, $B$, and $C$ are the propositional letters satisfied in the states. The formula $E_3 A_2 B$ holds in Model 2.3(a), but not in 2.3(b).

Given a FOC
CoRe automaton $H$, its corresponding untimed model $M_H$, $P$ (see Definition 3.2.5), and a formula $Q \in \Phi_P$, we can decide $M_H, P, \langle v, r \rangle \models Q$ by reducing the problem to a first-order formula validity problem as follows.

**Definition 5.4.1** Let $H$ be a FOC
CoRe, $Q$ be a formula of $\Phi_P$, and $v$ be a state of $H$. We define the formula $\varrho(H, Q, v)[Z]$ by induction on $Q$ as follows:

- $\varrho(H, P[Z], v)[Z]$ is $\text{Inv}(v)[Z] \land P[Z];$
- $\varrho(H, \neg P[Z], v)[Z]$ is $\text{Inv}(v)[Z] \land \neg P[Z];$
- $\varrho(H, Q_1 \lor Q_2, v)[Z]$ is $\varrho(H, Q_1, v)[Z] \lor \varrho(H, Q_2, v)[Z];$
- $\varrho(H, E_3 Q_1, v)[Z]$ is
  $$\bigvee_{ph \in \mathcal{P}_E(v)} (\exists Z' \exists T \geq 0 \widehat{\text{Reach}}(H, ph)[Z, Z', T] \land \varrho(H, Q_1, u_{ph})[Z]);$$
- $\varrho(H, A_2 Q_1, v)[Z]$ is
  $$\bigwedge_{ph \in \mathcal{P}_E(v)} (\forall Z' \exists T \geq 0 \widehat{\text{Reach}}(H, ph)[Z, Z', T] \rightarrow \varrho(H, Q_1, u_{ph})[Z']);$$

where we use $u_{ph} \in \mathcal{V}$ to denote the last node of $ph \in \mathcal{P}_E(v)$.

The following theorem associates the validity of the first-order formula $\varrho(H, Q, v)$ with the $\Phi_P$-formula $Q$.

**Theorem 5.4.2** Let $H$ be a FOC
CoRe automaton, $Q$ be a formula of $\Phi_P$, and $M_{H, P}$ be the untimed corresponding model of $H$. It holds that:

$$\langle M_{H, P}, \langle v, r \rangle \models Q \text{ if and only if } \varrho(H, Q, v)[r] \text{ holds.}$$

**Proof.** We proceed by structural induction on $Q$. The only interesting cases are the formulæ $E_3 Q_1$ and $A_2 Q_1$. We prove the statement in the case $E_3 Q_1$, since the other case has a similar proof.
If the automaton \( \varrho \), \( \langle v', s \rangle \) reachable from \( \langle v, r \rangle \), it holds that \( \langle v', s \rangle \Rightarrow Q_1 \). But, by Lemma 5.2.1, we can deduce that \( \langle v', s \rangle \) is reachable from \( \langle v, r \rangle \) if and only if there exists a \( ph \in T_E(v) \) such that \( \exists T \geq 0 \text{Reach}(H, ph)[r, s, T] \) holds and \( v' = u_{ph} \). Moreover, by \( H \)'s semantics, if \( \varrho (H, E \diamond Q_1, v) [r] \) holds, then, if and only if \( \varrho (H, E \diamond Q_1, v) [r] \) holds.

\( \Rightarrow \) If \( \varrho (H, E \diamond Q_1, v) [r] \) is true, then one of its disjoint holds. Let \( ph \) be the path whose disjoint holds. By Lemma 5.2.1, we can deduce that if the formula \( \exists T \geq 0 \text{Reach}(H, ph)[r, s, T] \) holds, and \( ph \in T_E(v) \), then \( \langle u_{ph}, s \rangle \) is reachable from \( \langle v, r \rangle \). Moreover, by inductive hypothesis, \( \langle u_{ph}, Z \rangle \Rightarrow Q_1 \) holds if and only if \( \varrho (H, Q_1, u_{ph}) [Z] \) holds. Hence, by \( \varphi_H \)'s semantics, if \( \varrho (H, E \diamond Q_1, v) [r] \) holds, then \( \langle u_{ph}, s \rangle \) is reachable from \( \langle v, r \rangle \) and \( \langle M_H, \varphi \rangle, \langle v, r \rangle \Rightarrow Q_1 \). It follows that \( \langle M_H, \varphi \rangle, \langle v, r \rangle \Rightarrow E \diamond Q_1 \).

Moreover, we can give some partial results over \( \varphi_H \) extended with the operator \( \sqcup \). Consider the following grammar obtained from \( \varphi_H \) by adding such operator.

\[
Q ::= P[Z] \mid \neg P[Z] \mid Q_1 \lor Q_2 \mid E \diamond Q_1 \mid \sqcup Q_1 \mid E(Q_1 \sqcup Q_2)
\]

Such language, which is called \( \varphi_H \), satisfies the following results.

**Theorem 5.4.3** Let \( H = \langle Z, Z', \nu, \xi, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset} \rangle \) be a \( \text{FOCoRe} \) and \( v \in \nu \) be a location of \( H \). Moreover, let \( Q_1 \) and \( Q_2 \) be two formulae of \( \varphi_H \) and \( H' \) be the hybrid automaton \( \hat{H} = \langle Z, Z', \nu, \xi, \text{Inv}', \text{Dyn}, \text{Act}, \text{Reset} \rangle \), where the invariants \( \text{Inv}' \) are defined as \( \text{Inv}'(v)[Z] \overset{\text{def}}{=} \text{Inv}(v)[Z] \land \varrho (H, Q_1, v) [Z] \lor \varrho (H, Q_2, v) [Z] \) for all \( v \in \nu \). Consider the formula \( \varphi (H, H', E(Q_1 \sqcup Q_2), v) [Z] \) defined by

\[
\varphi (H, H', E(Q_1 \sqcup Q_2), v) [Z] \overset{\text{def}}{=} \left( \exists T \geq 0 \exists Z' \bigvee_{ph \in T_E(v)} \text{Reach}(H', ph)[Z, Z', T] \land \varrho (H, Q_2, u_{ph}) [Z'] \right).
\]

If the automaton \( H' \) is a \( \text{FOCoRe} \) and the formula \( \varphi (H, H', E(Q_1 \sqcup Q_2), v) [r] \) holds, then \( \langle M_H, \varphi \rangle, \langle v, r \rangle \Rightarrow E(Q_1 \sqcup Q_2) \).

**Proof.** By Lemma 4.3.3 and by \( \text{Reach} \)'s definition, if \( H' \) is a \( \text{FOCoRe} \), then the formula \( \text{Reach}(H', ph)[p, q, t] \) holds if and only if \( H' \) reaches \( q \) from \( p \) in time \( t \) through a trace whose corresponding path is \( ph \). Moreover, by hypothesis, \( \text{Inv}'(v)[Z] \overset{\text{def}}{=} \text{Inv}(v)[Z] \land \varrho (H, Q_1, v) [Z] \lor \varrho (H, Q_2, v) [Z] \) for all \( v \in \nu \). Hence if the formula
\( \widehat{\text{Reach}}(H', \phi_h)[p, q, t] \) holds, then all the points in the trajectory from \( p \) to \( q \) satisfy either \( \varrho(H, Q_1, v)[Z] \) or \( \varrho(H, Q_2, v)[Z] \). Furthermore, since \( H \) and \( H' \) have the same dynamics, activations, and resets, if the formula \( \widehat{\text{Reach}}(H', \phi_h)[p, q, t] \) holds, then \( \widehat{\text{Reach}}(H, \phi_h)[p, q, t] \) holds too. Now consider the formula \( \bar{\varrho}(H, H', E(Q_1 \cup Q_2), v)[Z] \). If \( \bar{\varrho}(H, H', E(Q_1 \cup Q_2), v)[r] \) holds, then

\[
\exists T \geq 0 \exists Z' \forall p' \in \overline{T}_E(v) \left( \right.
\begin{array}{c}
\widehat{\text{Reach}}(H', \phi_h)[r, Z', T] \land \varrho(H, Q_2, u_{\phi_h})[Z']
\end{array}
\]

holds too. By above considerations, it follows that \( H \) can reach from \( r \) a point \( p \) satisfying \( \varrho(H, Q_2, u_{\phi_h})[p] \) through a trajectory in which either \( \varrho(H, Q_1, u_{\phi_h})[Z] \) or \( \varrho(H, Q_2, u_{\phi_h})[Z] \) holds. Thus \( (\mathcal{M}_{H, \pi}, (v, r) \models E(Q_1 \cup Q_2) \) by Theorem 5.4.2 and by \( \Phi_{H, \pi} \) semantics.

**Theorem 5.4.4** Let \( H = (Z, Z', \varphi, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset}) \) and \( v \in \mathcal{V} \) be a \( H \)'s location. Moreover, let \( Q_1 \) and \( Q_2 \) be two \( \Phi_{H, \pi} \) formulae and \( H' \) be the hybrid automaton \( (Z, Z', \varphi, E, \text{Inv}, \text{Dyn}, \text{Act}, \text{Reset}) \) where \( \text{Inv}'(v)[Z] \) \( \text{def} \) \( \text{Inv}(v)[Z] \land \varrho(H, Q_1, v)[Z] \) for all \( v \in \mathcal{V} \). Consider the formula \( \bar{\varrho}(H, H', E(Q_1 \cup Q_2), v)[Z] \) defined by

\[
\exists Z' \exists T \geq 0 \left( \begin{array}{c}
\forall 0 \leq T' < T \exists Z''
\end{array}
\right.
\begin{array}{c}
\forall p' \in \overline{T}_E(v) \forall p'' \in \overline{T}_E(u_{\phi_h})
\left(\right.
\begin{array}{c}
\widehat{\text{Reach}}(H', \phi_h)[Z, Z'', T'] \land \\
\widehat{\text{Reach}}(H, \phi_h')[Z'', Z', T - T'] \land \varrho(H, Q_2, u_{\phi_h'})[Z']
\end{array}
\right)
\end{array}
\left.
\begin{array}{c}
\forall
\end{array}
\right.
\begin{array}{c}
\exists T' > 0 \forall 0 < T'' \leq T' \exists Z''
\end{array}
\begin{array}{c}
\forall p' \in \overline{T}_E(v) \forall p'' \in \overline{T}_E(u_{\phi_h})
\left(\right.
\begin{array}{c}
\widehat{\text{Reach}}(H', \phi_h)[Z, Z', T] \land \\
\widehat{\text{Reach}}(H, \phi_h')[Z', Z'', T'' \land \varrho(H, Q_2, u_{\phi_h'})[Z'']]
\end{array}
\right)
\end{array}
\]

where we use \( u_p \in \mathcal{V} \) to denote the last node of a path \( p \). If \( H \) and \( H' \) are FO-CoRe, Dyn is transitive, and \( \bar{\varrho}(H, H', E(Q_1 \cup Q_2), v)[q] \) holds, then \( (\mathcal{M}_{H, \pi}, (v, q) \models E(Q_1 \cup Q_2) \).}

**Proof.** If \( \bar{\varrho}(H, H', E(Q_1 \cup Q_2), v)[q] \) holds, then there exist two paths \( ph \in \overline{T}_E(v) \) and
\(ph' \in T_E(u_{ph})\) such that either
\[
\phi_1 \overset{\text{def}}{=} \exists Z' \exists T \geq 0 \left( \forall 0 \leq T' < T \ \exists Z'' \left( \widehat{\text{Reach}}(H', ph)[Z, Z', T'] \land \widehat{\text{Reach}}(H, ph')[Z'', T - T'] \land \varrho(H, Q_2, u_{ph'})[Z'] \right) \right)
\]
or
\[
\phi_2 \overset{\text{def}}{=} \exists Z' \exists T \geq 0 \left( \exists T'' > 0 \ \forall 0 < T'' \leq T' \ \exists Z'' \left( \widehat{\text{Reach}}(H', ph)[Z, Z', T] \land \widehat{\text{Reach}}(H, ph')[Z'', T'] \land \varrho(H, Q_2, u_{ph'})[Z'''] \right) \right)
\]
holds. If \(\phi_1\) holds, then there exist a \(Z'\) and a \(T \geq 0\) such that for all \(T' \in [0, T)\) the formula
\[
\exists Z'' \left( \widehat{\text{Reach}}(H', ph)[Z, Z'', T'] \land \widehat{\text{Reach}}(H, ph')[Z', Z'', T - T'] \land \varrho(H, Q_2, u_{ph'})[Z'] \right)
\]
holds too. Hence, since \(Dyn\) is transitive dynamics, if \(\phi_1\) holds, then there exist a \(Z'\) and a \(T \geq 0\) such that the formula \(\widehat{\text{Reach}}(H', ph \cdot ph')[Z, Z', T] \land \varrho(H, Q_2, u_{ph \cdot ph'})[Z']\) holds by Lemma 4.3.5. Thus, by Lemma 4.3.3 and by \(\text{Reach}\)'s definition, \(H'\) can reach the state \(\langle u_{ph \cdot ph'}, Z' \rangle\) from \(\langle v, Z \rangle\) and \(\langle M_{H, p}, \rho \rangle, \langle u_{ph \cdot ph'}, Z' \rangle \models Q_2\) by Theorem 5.4.2. Moreover, if \(\phi_1\) holds, we can deduce that for all times \(T' \in [0, T)\) there exist a \(Z''\), a \(H'\)'s trajectory, \((\rho_i)_{i \in I}\), from \(\langle v, Z \rangle\) to \(\langle u_{ph \cdot ph'}, Z' \rangle\) passing through \(\langle u_{ph}, Z'' \rangle\), a \(H'\)'s trajectory, \((\rho'_i)_{i \in I'}\), from \(\langle v, Z \rangle\) to \(\langle u_{ph}, Z'' \rangle\) such that \(\rho_j = \rho'_i\) for all \(j < 7\), \(\rho_7'(t) = \rho_7(t)\) for all \(t \in [0, 7]\), and \(T' = \tilde{t} + \sum_{j=0}^{7} \text{Dom}(\rho_j)\), and \(\rho_7'(t) = \langle u_{ph}, Z'' \rangle\). For all \(i \in I'\) and for all \(t \in \text{Dom}(\rho'_i)\), \(\langle M_{H, p}, \rho(t) \models Q_1 \rangle\) by \(H'\)'s definition and by Theorem 5.4.2. It follows that \(\langle M_{H, p}, \langle v, q \rangle \models E Q_1 \cup Q_2 \rangle\). In an analogous way, we can prove the same result if \(\phi_2\) holds.

Despite the above results do not guarantee the decidability of \(\Phi_{U, p}\), they give us sufficient conditions to prove the existence of a trajectory \((\rho_i)_{i \in I}\) leaving a state \(\langle v, r \rangle\) such that the properties \(Q_2\) holds on \((\rho_i)_{i \in I}\) until the \(Q_2\) does. Verifying such existence is crucial in safety verification, when we require that a property does not happen until some security states have been reached. For these reasons, we think that, even if both Theorem 5.4.3 and Theorem 5.4.4 do not present an algorithm for deciding \(\langle M_{H, p}, \langle v, q \rangle \models E Q_1 \cup Q_2 \rangle\), they are interesting results in safety verification of FOCoRe.
Independent Dynamics Hybrid Automata

“O-minimal dynamical systems has gained some interest few years ago in the community, mostly due to the possibility to associate one’s name with that of Tarski and others.”

Unknown reviewer

In the previous chapter, we presented a class of hybrid automata, FOCoRe, for which, exploiting the constant reset condition, we can reduce reachability problem to a decidability problem for first-order formulæ. Even if such result is quite interesting from a theoretical point of view, the constant reset condition restricts FOCoRe applicability. For these reasons, we aim to relax such a condition. In this chapter, we identify a hybrid automaton class whose resets are not constant and we prove that for such class reachability problem can be reduced to a decidability problem for first-order formulæ. To achieve these results we restrict the allowed dynamics. In particular, we require that, if a reset is not constant on a variable, then the dynamics of such variable both do not change from location to location and do not depend on other variables. This means that we can distinguish two sets of variables: the first set includes all the variables whose resets are not constant, while the second includes only constant reset variables. Since the dynamics of variables in the former set should be independent from the values of variables in the latter, we call the hybrid automaton class presented in this chapter Independent Dynamics Automata or IDA.

6.1 Basic Notions

In this section we extend some of the concepts introduced in Section 2.1. In particular, we present two new notions: the improper subgraph and the touching path component.

Given a directed graph, an improper subgraph is obtained by considering a subset of nodes and some of the incident edges.
Definition 6.1.1 (Improper Subgraph) A graph \( G' = (V', E') \) is an improper subgraph of \( G \) if and only if \( V' \subseteq V \) and \( E' \subseteq E \cap (V' \times V') \).

Remark 6.1.2 Notice that, given a graph \( G \), there may be \( G \)'s improper subgraphs which are not \( G \)'s subgraphs. For instance, let us consider the two directed graphs \( G = (V, E) \) and \( G' = (V', E') \), where \( V = V' = \{v_1, v_2, v_3\} \), \( E = \{(v_1, v_2), (v_2, v_3), (v_1, v_3)\} \), and \( E' = \{(v_1, v_2), (v_2, v_3)\} \). Since \( E \cap (V' \times V') = E \neq E' \), it is easy to verify that \( G' \) is a \( G \)'s improper subgraph, but it is not a \( G \)'s subgraph.

A touching path component is an improper subgraph closed with respect to the relation subsequent.

Definition 6.1.3 (Touching Path Component) Let \( G = (V, E) \) be a graph. A graph \( T \) is a touching path component of \( G \) if and only if:

1. \( T = (V', E') \) is an improper subgraph of \( G \);
2. if \( e \) and \( e' \) are subsequent in \( G \), then \( e \in E' \) if and only if \( e' \in E' \);
3. if \( G' \) is an improper subgraph of \( G \) included in \( T \), then \( G' \) does not satisfy the condition 2.

Example 6.1.4 The graph \( G = (V, E) \) presented in Figure 6.1 has two touching path components \( T_0 = (V_0, E_0) \) and \( T_1 = (V_1, E_1) \). In particular, \( V_0 = \{v_0, v_1, v_2, v_4\} \) and \( E_0 = \{e_1, e_2, e_3, e_5\} \) while \( V_1 = \{v_0, v_3\} \) and \( E_1 = \{e_4\} \).

![Figure 6.1: Two different touching path components.](image)

Touching path components naturally induce a partition on the edges of a graph. The main property of touching path components, which we will exploit later, is that the edges occurring in a path always belong to the same path component.

Lemma 6.1.5 The edges \( e \) and \( e' \) belong to the same touching path components if and only if there exists a sequence \( e_1, e_2, \ldots, e_n \) such that \( e_1 = e, e_n = e' \) and for each \( i \in [1, n - 1] \) the edges \( e_i \) and \( e_{i+1} \) are subsequent.

Proof. If \( G = (V, E) \) is a graph any touching path component \( T \) of \( G \) is an improper subgraph of \( G \).

\( (\Rightarrow) \) Let us assume that \( e \) and \( e' \) belong to the same touching path component. Consider the sequence \( E_1(e), E_2(e), \ldots, E_k(e) \) such that:
• $\mathcal{E}_1(e) = \{e\};$

• $\mathcal{E}_i(e) = \mathcal{E}_{i-1}(e) \cup \{e'' | \exists e'' \in \mathcal{E}_{i-1}(e) \text{ such that } e'' \text{ and } e''' \text{ are subsequent}\};$

• $\mathcal{E}_h(e) = \mathcal{E}_{h-1}(e).$

We prove that there exists a sequence $e_1, e_2, \ldots, e_n$ such that $e_1 = e$, $e_n = e'$ and for each $j \in [1, n - 1]$ the edges $e_j$ and $e_{j+1}$ are subsequent by induction on $n$. If $n = 1$, then $e = e'$ and the thesis holds for the sequence $e$. If $n > 1$, then $e'$ has been added to $\mathcal{E}_n(e)$ since there exists $e'' \in \mathcal{E}_{n-1}(e)$ such that $e''$ and $e'$ are subsequent. Hence, since by inductive hypothesis we have proved the thesis on $n - 1$, there exists a sequence of the form $e_1, e_2, \ldots, e_{n-1}$, with $e_1 = e$, $e_{n-1} = e''$ and for each $j \in [1, n-2]$ the edges $e_j$ and $e_{j+1}$ are subsequent. We get that there exists a sequence $e_1, e_2, \ldots, e_{n-1}, e_n$ such that $e_1 = e$, $e_n = e'$ and for each $i \in [1, n-1]$ the edges $e_i$ and $e_{i+1}$ are subsequent.

($\Rightarrow$) Let us assume that there exists an edge sequence $e_1, e_2, \ldots, e_{n-1}, e_n$ such that $e_1, e_n$ and for each $i \in [1, n-1]$ the edges $e_i$ and $e_{i+1}$ are subsequent. We now prove that $e_1$ and $e_n$ belong to the same touching path components by induction on $n$. If $n = 1$, then $e_1 = e_n$ and the thesis holds. If $n = 2$, then $e_1$ and $e_n$ are subsequent edges, then, by Definition 6.1.3, $e_1$ belongs to a touching path $\mathcal{T}$ if and only if $e_n$ belongs $\mathcal{T}$. The thesis follows directly. If $n > 1$, then by inductive hypothesis $e_1$ and $e_{n-1}$ belong to the same touching path components. Since $e_{n-1}$ and $e_n$ are subsequent by hypothesis, $e_{n-1}$ belongs to a touching path $\mathcal{T}$ if and only if $e_n$ belongs $\mathcal{T}$. It follows that $e_1$ and $e_n$ belong to the same touching path components. 

Lemma 6.1.6 Any graph $G$ has a finite number of touching path components which are connected and partition the set of edges, i.e., each edge belongs to exactly one touching path component.

Proof. If $G = (\mathcal{V}, \mathcal{E})$ is a graph any touching path component $\mathcal{T}$ of $G$ is an improper subgraph of $G$. Thus if $\mathcal{T} = (\mathcal{V'}, \mathcal{E'})$ then $\mathcal{V'} \subseteq \mathcal{V}$ and $\mathcal{E'} \subseteq \mathcal{E} \cap (\mathcal{V'} \times \mathcal{V'})$. Hence, since the number of subsets of a set $S$ is $2^{|S|}$, the number of touching path components of $G$ are upper bounded by $2^{|\mathcal{V}|+|\mathcal{V}'|}$. The third condition of the definition of touching path component ensures that $\mathcal{T}$ is connected.

If $\mathcal{T}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ and $\mathcal{T}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ are two different touching path components, either $\mathcal{E}_0 \subseteq \mathcal{E}_1$, $\mathcal{E}_0 \supseteq \mathcal{E}_1$, $\mathcal{E}_0 = \mathcal{E}_1$, $\mathcal{E}_0 \cap \mathcal{E}_1 = \emptyset$ or $\mathcal{E}_0 \cap \mathcal{E}_1 \neq \emptyset$. Assume, by contradiction, that $\mathcal{E}_0 \nsubseteq \mathcal{E}_1$. Hence there exists an edge $e \in \mathcal{E}$ such that $e \in \mathcal{E}_0$ and $e \notin \mathcal{E}_1$. In this case, there should exist an edge $e' \in \mathcal{E}$ such that $e' \in \mathcal{E}_0$ and $e' \notin \mathcal{E}_1$. By Lemma 6.1.5, $e, e' \in \mathcal{E}_0$ if and only if there exists a sequence of $G$’s edges $e_0, \ldots, e_n$ such that $e_i$ and $e_{i+1}$ are subsequent for all $i \in [0, n - 1]$ and either $e = e_0$ and $e' = e_n$ or $e = e_n$ and $e' = e_0$. Since $e \in \mathcal{E}_1$ by hypothesis, it follows that $e' \in \mathcal{E}_1$ by the same definition. But this contradicts the hypothesis thus if $\mathcal{T}_0 \neq \mathcal{T}_1$ then $\mathcal{E}_0 \nsubseteq \mathcal{E}_1$. In a similar way, we can prove that $\mathcal{E}_0 \nsubseteq \mathcal{E}_1$. Moreover, since $\mathcal{T}_0$ and $\mathcal{T}_1$ are connected, it follows that if $\mathcal{T}_0 \neq \mathcal{T}_1$, then $\mathcal{E}_0 = \mathcal{E}_1$ does not hold. Let us assume that $\mathcal{E}_0 \supseteq \mathcal{E}_0 \cap \mathcal{E}_1 \subset \mathcal{E}_1$ and $\mathcal{E}_0 \cap \mathcal{E}_1 \neq \emptyset$. In such case, there exist $\overline{e_0}, \overline{e_1}$, and $\overline{e}$ such that $\overline{e_0} \in \mathcal{E}_0 \setminus \mathcal{E}_1$, $\overline{e_1} \in \mathcal{E}_1 \setminus \mathcal{E}_0$, and $\overline{e} \in \mathcal{E}_0 \cap \mathcal{E}_1$. By Lemma 6.1.5, it follows that there exist two sequences of edges $e'_1, \ldots, e'_{n'}$ and $e''_1, \ldots, e''_{n''}$ such that $e'_1 = \overline{e_0}$, $e'_{n'} = \overline{e}$, $e''_1 = \overline{e'}$, $e''_{n''} = \overline{e''}$, for all $i \in [1, n' - 1]$, $e'_i$ and
$e_{i}^{r+1}$ are subsequent, and for all $i \in [1, n'-1]$, $e_{i}^{r}$ and $e_{i+1}^{r}$ are subsequent. Hence, there exists a sequence of edges $\tilde{e}_{1}, \ldots, \tilde{e}_{b}$ such that $\tilde{e}_{i} = \tilde{e}_{i}^{r}$, $\tilde{e}_{i+1}^{r}$, and for all $i \in [1, \pi - 1]$, $\tilde{e}_{i}$ and $\tilde{e}_{i+1}$ are subsequent. Thus, by Lemma 6.1.5, $\tilde{e}_{0}$ and $\tilde{e}_{1}$ belong to the same touching path components and $E_{0} = E_{1}$. But this contradicts the hypothesis if $T_{0} \neq T_{1}$, then $E_{0} \cap E_{1} = \emptyset$. Obviously for every $e \in E$ there exists a touching path component including it.

Given a trace of a hybrid automaton $H$, the corresponding path of such trace is a path of the discrete component of $H$. Hence the corresponding path’s edges belong to the same touching path component.

**Lemma 6.1.7** Let $H = (Z, Z', V, E, Inv, Dyn, Act, Reset)$ be a hybrid automaton and $tr$ be a trace of $H$. The corresponding path $ph$ of $tr$ is included into a touching path component of the graph $(V, E)$.

**Proof.** This is an immediate consequence of Lemma 6.1.5.

Moreover, from Lemma 6.1.6 and Lemma 6.1.5, we can deduce an algorithm to compute the touching path components of a directed graph $G$.

**Algorithm 3** Compute the touching path components of a directed graph $G$

**Require:** A directed graph $G = (V, E)$

**Ensure:** Return the set, $S$, of the touching path components of $G$.

$S \leftarrow \emptyset$

for all $v \in V$ do {All locations belong to a touching path components}

$S \leftarrow S \cup \{(\{v\}, \emptyset)\}$

end for

for all $(v, v') \in E$ do {For each edge $e = (v, v')$}

$V_{0} \leftarrow \{v, v'\}$

$E_{0} \leftarrow \{(v, v')\}$

{If there exists an edge $e' \in (V_{1}, E_{1})$ such that $e$ and $e'$ are subsequent, then they belong to the same touching path component.}

for all $(V_{1}, E_{1}) \in S$ such that $\exists (v'', v) \in E_{1}$ or $\exists (v', v''') \in E_{1}$ or $E_{1} = \emptyset$ and $\forall v \in V_{1} \forall v' \in V_{1}$ do

$S \leftarrow S \setminus \{(V_{1}, E_{1})\}$ {Remove $(V_{1}, E_{1})$ from $S$}

$V_{0} \leftarrow V_{0} \cup V_{1}$ {Expand the touching path component of $e$ by adding the locations and the edges of $(V_{1}, E_{1})$}

$E_{0} \leftarrow E_{0} \cup E_{1}$

end for

$S \leftarrow S \cup \{(V_{0}, E_{0})\}$ {Add the new touching path component of $e$ to $S$}

end for

Return $S$

The termination of Algorithm 3 is guaranteed by the finiteness of $G$’s edges, while the correctness follows directly from Lemma 6.1.6 and Lemma 6.1.5.
6.2 A New Hybrid Automaton Class

In this section, we define a new class of hybrid automata. Edges of automata belonging to such class can have variables which are not reset, i.e., they are reset using the identity function. On these variables we need to have the same dynamics before and after edge crossing. This condition is similar to that used in rectangular initialised hybrid automata (see [81, 87]). To distinguish the variables which are not reset from the remaining ones we need to partition the set of variables. In this partition there are always two classes and one of them could be empty. Moreover, a partition of the variables occurring in the vector $Z$ naturally induce a partition on the variables occurring in $Z'$.

**Definition 6.2.1 (Induced Partition)** Let $H$ be a hybrid automaton. Moreover, let $P = \{I, D\}$ be a partition of $\Gamma_Z$. The partition $P' = \{I', D'\}$ such that for each $i \in [0, k]$, $Z_i \in I$ if and only if $Z'_i \in I'$ is the partition induced by $P$ on $\Gamma_{Z'}$.

**Remark 6.2.2** Notice that, if $P'$ is induced by $P = \{I, D\}$, then both $P$ and $P'$ are uniquely determined by $I$.

We now introduce the notion of f-independent variables where the “f” stands for “in the future”. A subset $I$ of the variables occurring in $Z$ is f-independent from the other variables occurring in $Z$, denoted by $D$, if their future behaviour does not depend on the variables of $D$.

**Definition 6.2.3 (F-Independent Variables)** Let $H$ be a hybrid automaton, and $\psi[Z, Z', T]$ be a formula. Moreover, let $P = \{I, D\}$ be a partition of $\Gamma_Z$ and $P' = \{I', D'\}$ be the partition induced by $P$ on $\Gamma_{Z'}$. If $\psi$ is equivalent to $\exists Z D \exists Z' \psi \land (\exists Z' \psi)$, then we say that $I$ is f-independent from $D$ on $\psi$. In this case we also say that $I$ is f-independent from on $\psi$, that $\rho(\psi, I) \overset{\text{def}}{=} \exists Z D \exists Z' \psi$ is the ruling formula for $I$ on $\psi$ and that $\sigma(\psi, I) \overset{\text{def}}{=} \exists Z' \psi$ is the unrelated formula for $I$ on $\psi$.

**Remark 6.2.4** Since $\rho(\psi, I) \overset{\text{def}}{=} \exists Z D \exists Z' \psi$, all the free variables in $\rho(\psi, I)$ are included in $I \cup I' \cup \{T\}$. Analogously, all the free variables in $\sigma(\psi, I)$ are included in $I \cup D \cup D' \cup \{T\}$.

We have now to instantiate the notion of f-independent variables to the case in which we are considering a location $v$ of $H$. In this case we have to consider the formula $\text{Dyn}(v)[Z, Z', T]$.

**Definition 6.2.5 (F-Independent on Location)** Let $H$ be a hybrid automaton, $P = \{I, D\}$ be a partition of $\Gamma_Z$ and $v \in V$ be a location of $H$. The variable set $I$ is f-independent on $v$ if and only if $I$ is f-independent on $\text{Dyn}(v)[Z, Z', T]$.

In this case we call ruling formula for $I$ on $v$ the ruling formula $\rho(\text{Dyn}(v), I)$ and unrelated formula for $I$ on $v$ the unrelated formula $\sigma(\text{Dyn}(v), I)$.
Remark 6.2.6 Notice that given a location \( v \) of a hybrid automaton \( H \) there are many sets which are f-independent on \( v \). In particular, if \( I \) is f-independent on \( v \), then any subset of \( I \) is also f-independent on \( v \).

Example 6.2.7 Let \( v \) be a location such that the formula \( \text{Dyn}(v)[Z, Z', T] \) is equivalent to \( Z'_1 = 2T + Z_1 \wedge Z'_2 = T - Z_2 \). All the sets \( \emptyset, \{Z_1\}, \{Z_2\} \) and \( \{Z_1, Z_2\} \) are f-independent on \( v \).

When we focus our attention on an edge in the notion of f-independence we have to consider the reset formula.

Definition 6.2.8 (F-Independent on Edge) Let \( H \) be a hybrid automaton and \( \mathcal{P} = \{I, D\} \) be a partition of \( \Gamma_Z \). Moreover, let \( e \in \mathcal{E} \) be an edge of \( H \). The variable set \( I \) is said to be f-independent on \( e \) if and only if \( I \) is f-independent on \( \text{Reset}(e)[Z, Z'] \).

In this case we call ruling formula for \( I \) on \( e \) the formula \( \rho(\text{Reset}(e), I) \) and unrelated formula for \( I \) on \( e \) the formula \( \sigma(\text{Reset}(e), I) \).

Example 6.2.9 Let \( e \) be an edge such that \( \text{Reset}(e)[Z, Z'] \) is the formula \( Z'_1 = 8 \wedge Z'_2 = Z_1 \). The variable sets \( \emptyset \) and \( \{Z_1, Z_2\} \) are obviously f-independent on \( e \). Moreover, since \( \rho(\text{Reset}(e), \{Z_1\}) \equiv \exists Z_2 \exists Z'_1(Z'_1 = 8 \wedge Z'_2 = Z_1) \equiv Z'_1 = 8 \) and \( \sigma(\text{Reset}(e), \{Z_1\}) \equiv \exists Z'_1(Z'_1 = 8 \wedge Z'_2 = Z_1) \equiv Z'_2 = Z_1 \), \( \text{Reset}(e)[Z, Z'] \) is equivalent to \( \rho(\text{Reset}(e), \{Z_1\}) \wedge \sigma(\text{Reset}(e), \{Z_1\}) \equiv Z'_1 = 8 \wedge Z'_2 = Z_1 \). It follows that \( \{Z_1\} \) is f-independent on \( e \). On the contrary, the set \( \{Z_2\} \) is not f-independent on \( e \). As a matter of fact, if \( \{Z_2\} \) was f-independent on \( e \), then the formula \( \text{Reset}(e)[Z, Z'] \) is equivalent to \( \rho(\text{Reset}(e), \{Z_2\}) \wedge \sigma(\text{Reset}(e), \{Z_2\}) \). However, \( \rho(\text{Reset}(e), \{Z_2\}) \equiv \exists Z_1 \exists Z'_1(Z'_1 = 8 \wedge Z'_2 = Z_1) \) and, hence \( \rho(\text{Reset}(e), \{Z_2\}) \equiv \text{tt} \). Moreover, \( \sigma(\text{Reset}(e), \{Z_2\}) \equiv \exists Z'_1(Z'_1 = 8 \wedge Z'_2 = Z_1) \), and thus \( \sigma(\text{Reset}(e), \{Z_2\}) \equiv Z'_1 = 8 \). Obviously, \( Z'_1 = 8 \wedge \text{tt} \) and \( Z'_1 = 8 \wedge Z'_2 = Z_1 \) are not equivalent. It follows that \( \{Z_2\} \) is not f-independent on \( e \).

The following notion of persistent partition guarantees that the variables that are not reset on an edge \( e \) (i.e., the f-independents on \( e \)) have the same dynamics before and after crossing \( e \). Moreover, it is required that the f-independent variables on \( e \) do not recur in the resets of the remaining variables.

Definition 6.2.10 (Persistent Partition) Let \( H \) be a hybrid automaton. Moreover, let \( \mathcal{P} = \{I, D\} \) be a partition of \( \Gamma_Z \) and \( e = \langle v, v' \rangle \in \mathcal{E} \) be an edge of \( H \). The partition \( \mathcal{P} \) is a persistent partition of \( e \) if and only if we have that:

1. \( I \) is f-independent on \( e, v \), and \( v' \);
2. \( \rho(\text{Dyn}(v), I) \) and \( \rho(\text{Dyn}(v'), I) \) are equivalent;
3. \( \rho(\text{Reset}(e), I) \) and \( \bigwedge_{Z_i \in \mathcal{P}} Z_i = Z'_i \) are equivalent.

We are now ready to present our class of automata. In this class we mainly require that each edge has a persistent partition which is preserved on subsequent edges. Moreover, we impose that the invariants are closed and bounded and that
there is a minimum amount of time which has to be spent in a location between two jumps.

**Definition 6.2.11 (Independent Dynamics Automata)** A hybrid automaton $H$ is a
independent dynamics automaton, or simply a IDA, if:

1. $H$ is in Michael’s form;
2. Let $\text{MinDyn}(H, v, t, \tau)$ be the formula

\[
\text{MinDyn}(H, v, t, \tau) \overset{\text{def}}{=} \exists T > 0 \forall Z \forall Z' \forall T' \geq 0 \\
((v[Z] \wedge \tau[Z'] \wedge \text{Dyn}(v[Z, Z', T]) \rightarrow T' \geq T).
\]

The formula $\text{MinDyn}(H, v, \text{Reset}(e), \text{Act}(e'))$ holds for each pair of edges $e = (v, v')$ and $e' = (v, v'')$;
3. For all touching path component $T$ in $(V, E)$, there exists a partition $\mathcal{T}(T) = \{\mathcal{I}(T), \mathcal{D}(T)\}$ which is a persistent partition of each edge $e$ of $T$;
4. For all touching path component $T$ in $H$ and for all locations $v$ in $T$, the formula $\rho(\text{Dyn}(v), \mathcal{I}(T))$ is transitive.

**Example 6.2.12** Consider the hybrid automaton $H = (Z, Z', V, E, \text{Inv}, \mathcal{T}, \text{Act}, \text{Reset})$ such that:

- The dimension, $k$ of the automata is 2;
- The discrete projection $(V, E)$ is reported in Figure 6.2;
- The function $\text{Dyn}$ is such that:
  - $\text{Dyn}(v_1)[Z, Z', T] \equiv Z'_1 = 2(T)^3 + Z_1 \wedge Z'_2 = -3T + Z_2$;
  - $\text{Dyn}(v_2)[Z, Z', T] \equiv Z'_1 = -(T)^2 \wedge Z'_2 = -3T + Z_2$;
  - $\text{Dyn}(v_3)[Z, Z', T] \equiv Z'_1 = 2(T)^3 + Z_1 \wedge Z'_2 = -3T + Z_2$;
- The formulae $\text{Inv}(v_1)[Z]$, $\text{Inv}(v_2)[Z]$ and $\text{Inv}(v_3)[Z]$ are equal to $Z_2 \geq (Z_1)^2 \wedge Z_2 \leq 100$;
- The function $\text{Reset}$ is such that:
  - $\text{Reset}(e_1)[Z, Z'] \equiv Z'_1 \leq 8 \wedge Z'_2 = Z_2$;
  - $\text{Reset}(e_2)[Z, Z'] \equiv Z'_1 = Z_1 \wedge Z'_2 \geq 13$;
  - $\text{Reset}(e_3)[Z, Z'] \equiv Z'_1 = Z_1 \wedge Z'_2 > 7$;
- The function $\text{Act}$ is such that:
  - $\text{Act}(e_1)[Z] \equiv Z_2 > 5$;
  - $\text{Act}(e_2)[Z] \equiv Z_1 \wedge Z_2 \geq 0$;
Figure 6.2: The discrete component of an IDA. The complete definition of this automaton is reported in Example 6.2.12.

\[- \text{Act}(e_3)[Z] \equiv (Z_1)^2 + (Z_2 - 5)^2 \leq 8.\]

The automaton \( H \) is an IDA.

Since, by Lemma 6.1.6, each edge belongs to a single touching path component, if \( e \) is an edge in the touching path component \( \mathcal{T} \), then we may denote \( \mathcal{T}(\mathcal{T}) \) as \( \mathcal{T}(e) \), \( \mathcal{T}(\mathcal{D}) \) as \( \mathcal{D}(e) \), respectively. Moreover, we will use the following formulæ to denote the ruling and unrelated formula relative to the vertex \( v \) with respect to the persistent partition corresponding to the edge \( \langle v, v' \rangle \).

\[
\hat{\rho}(\langle v, v' \rangle)[Z, Z', T] \overset{\text{def}}{=} \rho(Dyn(v), I(\langle v, v' \rangle))[Z, Z', T]
\]

\[
\hat{\sigma}(\langle v, v' \rangle)[Z, Z', T] \overset{\text{def}}{=} \sigma(Dyn(v), I(\langle v, v' \rangle))[Z, Z', T]
\]

Moreover, we will denote the the ruling and unrelated formula on an edge using the following notations.

\[
\overline{\rho}(e)[Z, Z'] \overset{\text{def}}{=} \rho(\text{Reset}(e), I(e))[Z, Z']
\]

\[
\overline{\sigma}(e)[Z, Z'] \overset{\text{def}}{=} \sigma(\text{Reset}(e), I(e))[Z, Z']
\]

Notice that, if a touching path component \( \mathcal{T} \) is such that \( I(\mathcal{T}) = \emptyset \), then it can be studied applying the techniques presented in Chapter 4. Hence, from now on we do not consider this case.

### 6.3 Identifying an IDA

In this section, we show how to identify an IDA. In particular, we present a first-order formula which hold if and only if the considered hybrid automaton is an IDA.

By IDA’s definition, to decide whenever an automaton \( H = (V, E) \) is an IDA, we need to define a formula which:

- decides whenever \( H \) is in Michael’s form;
- checks if the formula \( \text{MinDyn}(H, v, \overline{\text{Reset}}(e), \text{Act}(e')) \) holds for each pair of edges \( e = \langle v', v \rangle \) and \( e' = \langle v, v'' \rangle \).
6.3. Identifying an IDA

- for each touching path component $T$ of $H$, verifies the existence of a partition $\mathcal{T}$ which is a persistent partition for all edges in $T$;

- characterises $\rho(Dyn(v), I(\mathcal{T}))$ transitivity, for each touching path component $T$ of $H$ and for all $T$'s location $v$.

By Section 4.2, a hybrid automaton $H$ is Michael’s form if and only if the formula $\bigwedge_{v \in V} MForm(H, v)$ holds. Moreover, since $|E|$ is finite, we can write the formula

$$\bigwedge_{\langle \langle v', v \rangle, \langle v, v'' \rangle \rangle \in Sb(H)} \text{MinDyn}(H, v, \text{Reset}(\langle v', v \rangle), \text{Act}(\langle v, v'' \rangle))$$

which, obviously, holds if and only if the formula $\text{MinDyn}(H, v, \text{Reset}(\langle v', v \rangle), \text{Act}(\langle v, v'' \rangle))$ holds for each pair of edges $e = \langle v', v \rangle$ and $e' = \langle v, v'' \rangle$.

By Definition 6.2.10, $P = \{I, D\}$ is a persistent partition of $\langle v, v' \rangle \in E$ if and only if

1. $I$ is f-independent on $\langle v, v' \rangle$, $v$, and $v'$;
2. $\rho(Dyn(v), I)$ and $\rho(Dyn(v'), I)$ are equivalent;
3. $\rho(\text{Reset}(e), I)$ and $\bigwedge_{Z_i \in I} Z_i = Z_i'$ are equivalent.

Moreover, by Definitions 6.2.3, $I$ is f-independent on $\phi[Z, Z', T]$ if and only if $\phi \equiv (\exists Z_{\phi'} \exists Z_{\phi'}' \phi) \land (\exists Z_{\phi'} \phi[Z, Z', T])$. Hence, we can write the formula

$$\text{FIndForm}(H, \mathcal{T}, \phi) \overset{\text{def}}{=} \forall Z \forall Z' \forall T \left( \phi[Z, Z', T] \iff ((\exists Z_{\phi'} \exists Z_{\phi'}' \phi[Z, Z', T]) \land (\exists Z_{\phi'} \phi[Z, Z', T])) \right)$$

which holds if and only if $I$ is f-independent on $\phi$. It follows that the formula

$$\text{TFIndForm}(H, \mathcal{T}, v, v') \overset{\text{def}}{=} \text{FIndForm}(H, \mathcal{T}, \text{Reset}(\langle v, v' \rangle)) \land \text{FIndForm}(H, \mathcal{T}, \text{Dyn}(v)) \land \text{FIndForm}(H, \mathcal{T}, \text{Dyn}(v'))$$

holds if and only if $I$ is f-independent on $\langle v, v' \rangle$, $v$, and $v'$. By $\rho$’s definition, $\rho(Dyn(v), I) \equiv \rho(Dyn(v'), I)$ if and only if the formula

$$\text{EquDynForm}(H, \mathcal{T}, v, v') \overset{\text{def}}{=} \forall Z \forall Z' \forall T \left( (\exists Z_{\phi} \exists Z_{\phi'} \text{Dyn}(v)[Z, Z', T]) \iff (\exists Z_{\phi} \exists Z_{\phi'} \text{Dyn}(v')[Z, Z', T]) \right)$$

holds. Furthermore, $\rho(\text{Reset}(e), I)$ and $\bigwedge_{Z_i \in I} Z_i = Z_i'$ are equivalent if and only if the formula

$$\text{IdResetForm}(H, \mathcal{T}, e) \overset{\text{def}}{=} \forall Z \forall Z' \left( (\exists Z_{\phi} \exists Z_{\phi'} \text{Reset}(e)[Z, Z']) \iff \right.$$


holds, while it is easy to see that \( \rho(\text{Dyn}(v), t) \)'s transitivity can be characterised by formula

\[
\text{TransForm}(H, \mathcal{P}, v) \overset{\text{def}}{=} \forall Z \forall Z' \forall Z'' \forall T \forall T' \left((\exists Z \exists Z' \exists Z'' \text{Dyn}(v)[Z, Z', T]) \land (\exists Z \exists Z' \exists Z'' \text{Dyn}(v)[Z', Z'', T']) \rightarrow (\exists Z \exists Z' \exists Z'' \text{Dyn}(v)[Z, Z'', T + T'])\right)
\]

Notice that we can compute the set \( S \) of all the touching path components of \( H \) using Algorithm 3. Moreover, \( S \) partitions \( \mathcal{E} \) and \(|S|, |\mathcal{E}| \) and the set \( P \) of all the possible partitions of \( \Gamma_Z \) are finite. It follows that we can write the formula

\[
\text{PPartTransForm}(H) \overset{\text{def}}{=} \bigwedge_{\langle \mathcal{V}', \mathcal{E}' \rangle \in S} \bigvee_{\langle \mathcal{V}, \mathcal{E} \rangle \in P} \left( \text{TFIndForm}(H, \mathcal{P}, v, v') \land \text{Equ DynForm}(H, \mathcal{P}, v, v') \land \text{TransForm}(H, \mathcal{P}, v) \land \text{IdResetForm}(H, \mathcal{P}, \langle v, v' \rangle) \right)
\]

which holds if and only if both Condition 3 and 4 of IDA’s definition hold. Thus the formula

\[
\text{IDAForm}(H) \overset{\text{def}}{=} \bigwedge_{\langle (v', v), (v, v'') \rangle \in S_b(H)} \text{MinDyn}(H, v, \overline{\text{Reset}}((v', v)), \overline{\text{Act}}((v, v''))) \land \bigwedge_{v \in \mathcal{V}} \text{MForm}(H, v) \land \text{PPartTransForm}(H)
\]

holds if and only if \( H \) is an IDA.

### 6.4 Reachability

In this section we show that reachability on IDA can be reduced to first-order formula decidability problem, provided that for the independent variables there is a maximum time at which a set is reachable. This assumption, paired with the fact that in IDA we have to spend a minimum amount of time in each location, allows us to compute the maximum length of the corresponding paths that we need to consider.

As we showed in Section 4.3 and recalled in Section 5.2, we can write a formula to decide reachability through a finite path. Unfortunately, if \( \langle \mathcal{V}, \mathcal{E} \rangle \) has a cycle, then it has an infinite number of paths. I the following sections, we will show that, under some assumptions, there exists a maximum length for paths that we need to consider to evaluate the reachability in IDA.
Thus:

\[ \overline{\rho}(e)[Z, Z'] \equiv \bigwedge_{Z_i \in I(e)} Z'_i = Z_i \]

Let \( W \subseteq V \) be a subset of locations. The following formula represents the invariants over the vertices of \( W \).

\[ S\text{Inv}(W)[Z] \equiv \bigvee_{v \in W} \text{Inv}(v)[Z] \]

Now we can prove that the continuous reachability implies the reachability in the projection over the independent variables inside a location. In particular, since the flow of the independent variables inside a location is expressed by the formula \( \rho(e)[Z, Z', T] \) we get the following result.

**Lemma 6.4.1** Let \( H \) be a IDA and \( W \) be a subset of \( V \). For each \( v \in W \) and each edge \( \langle v, v' \rangle \in E \) it holds that:

\[ \text{Reach}(H, v)[Z, Z', T] \Rightarrow \hat{\rho}(\langle v, v' \rangle)[Z, Z', T] \land \forall 0 \leq T' \leq T \exists Z'' (\hat{\rho}(\langle v, v' \rangle)[Z, Z'', T'] \land S\text{Inv}(W)[Z'']) \]

**Proof.** By definition of \( \text{Reach}(v) \), it holds that:

\[ \text{Reach}(H, v)[Z, Z', T] \equiv (\sim T > 0 \land \text{Dyn}(v)[Z, Z', T] \land Tp(v)[Z, T]) \lor \]

\[ (T = 0 \land Z = Z') \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'] \]

Since, by Definition 3.1.1, \( \text{Dyn}(v)[p, p, 0] \) holds for all locations \( v \) and for all \( p \), if \( T = 0, Z = Z' \), and \( \text{Inv}(v)[Z] \) hold, then \( \text{Dyn}(v)[Z, Z', T], Tp(v)[Z, T], \text{Inv}(v)[Z], \) and \( \text{Inv}(v)[Z'] \) hold too. Hence, by Definitions 6.2.10 and 6.2.5, it holds that:

\[ \text{Reach}(H, v)[Z, Z', T] \Rightarrow (\sim T > 0 \land \text{Dyn}(v)[Z, Z', T] \land Tp(v)[Z, T]) \lor \]

\[ (T = 0 \land Z = Z') \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'] \]

\[ \Rightarrow \text{Dyn}(v)[Z, Z', T] \land \]

\[ \forall 0 \leq T' \leq T \exists Z'' (\text{Dyn}(v)[Z, Z', T] \land \text{Inv}(v)[Z'']) \land \]

\[ \text{Inv}(v)[Z'] \land \text{Inv}(v)[Z] \]

\[ \Rightarrow (\hat{\rho}(e)[Z, Z', T] \land \hat{\sigma}(e)[Z, Z', T]) \land \]

\[ \forall 0 \leq T' \leq T \exists Z'' (\hat{\rho}(e)[Z, Z'', T] \land \hat{\sigma}(e)[Z, Z'', T] \land \]

\[ \text{Inv}(v)[Z'']) \land \text{Inv}(v)[Z] \land \text{Inv}(v)[Z'] \]

Thus:

\[ \text{Reach}(H, v)[Z, Z', T] \Rightarrow (\hat{\rho}(e)[Z, Z', T] \land \hat{\sigma}(e)[Z, Z', T]) \land \]
∀0 ≤ T' ≤ T ∃Z'' (\( \hat{\rho}(e)[Z, Z'', T] \) \( \wedge \) \( \hat{\sigma}(e)[Z, Z'', T] \) \( \wedge \) \( Inv(v)[Z''] \))

⇒ \( \hat{\rho}(e)[Z, Z', T] \) \( \wedge \) 

\( \forall 0 \leq T' \leq T \exists Z'' (\hat{\rho}(e)[Z, Z'', T] \wedge Inv(v)[Z'']) \)

Moreover, since \( v \in W \), if \( Inv(v)[Z''] \) holds then, the formula \( SInv(W)[Z''] \) holds by definition. Thus, if the formula \( Reach(H, v)[Z, Z', T] \) holds, then the formula \( \hat{\rho}(e)[Z, Z', T] \wedge \forall 0 < T' \leq T \exists Z'' (\hat{\rho}(e)[Z, Z'', T'] \wedge SInv(W)[Z'']) \) holds too.

Similarly, the discrete reachability over an edge \( e \) implies the formula \( \overline{\rho}(e) \), i.e., it implies that the independent variables do not change.

**Lemma 6.4.2** Let \( H \) be a IDA. For each edge \( e \in E \), the following holds:

\[
Reach(H, e)[Z, Z'] \Rightarrow \bigwedge_{Z_i \in \ell(e)} Z'_i = Z_i
\]

**Proof.** Let \( e = \langle v, v' \rangle \). By definition of \( Reach(H, e)[Z, Z'] \), it holds that:

\[
Reach(H, e)[Z, Z'] \equiv Inv(v)[Z] \wedge Act(e)[Z] \wedge \text{Reset}(e)[Z, Z'] \wedge Inv(v')[Z']
\]

Since \( H \) is in a IDA, by Definitions 6.2.10 and 6.2.8, it follows that:

\[
Reach(H, e)[Z, Z'] \equiv Inv(v)[Z] \wedge Act(e)[Z] \wedge \text{Reset}(e)[Z, Z'] \wedge Inv(v')[Z']
\]

But \( \overline{\rho}(e)[Z, Z'] \equiv \bigwedge_{Z_i \in \ell(e)} Z'_i = Z_i \) by definition. Thus, if \( Reach(H, e)[Z, Z'] \) holds, then \( \bigwedge_{Z_i \in \ell(e)} Z'_i = Z_i \) holds too.

Thanks to the two above lemmas we can prove that the reachability along a path projected on the independent variables corresponds to the reachability inside a location modulo an enlargement of the invariant.

**Lemma 6.4.3** Let \( H \) be a IDA, \( T = \langle V', E' \rangle \) be a touching path component of \( H \), and \( ph = \langle v_0, \ldots, v_h \rangle \) be a path in \( T \). The following holds:

\[
\widehat{Reach}(H, ph)[Z, Z', T] \Rightarrow \hat{\rho}(\langle v_0, v_1 \rangle)[Z, Z', T] \wedge \forall 0 \leq T' \leq T \exists Z'' (\hat{\rho}(\langle v_0, v_1 \rangle)[Z, Z'', T'] \wedge \overline{\rho}([V', T'])[Z''])
\]

where \( \widehat{Reach}(H, ph)[Z, Z', T] \) is the formula defined in Section 4.3.
Proof. By definition:
\[
\overline{\text{Reach}}(H, ph)[Z, Z', T] \equiv \exists Z^1, \ldots, Z^{2^h} \text{ Reach}(H, ph)[Z, Z^1, \ldots, Z^{2^h}, Z', T]
\]
\[
\equiv \exists Z^1, \ldots, Z^{2^h} \exists T_0 \geq 0 \ldots \exists T_h \geq 0 \left( T = \sum_{i=0}^{h} T_i \land \right.
\]
\[
\text{Reach}(H, v_0)[Z, Z^0, T_0] \land \text{Reach}(H, v_0)[Z^{2^h}, Z', T_h] \land \left( \text{Reach}(H, (v_i, v_{i+1}))[Z^{2^i}, Z^{2^{i+1}}, T_i] \land \right.
\]
\[
\left( \text{SInv}(V')[Z^{2^{i+1}}, Z^{2^{i+2}}] \right) \land \left( \text{SInv}(V')[Z^{4^{i+1}}, Z^{4^{i+2}}] \right) \land \left( \text{SInv}(V')[Z^{2^{i+2}}, Z^{2^{i+3}}, T_{i+1}] \land \right.
\]
\[
\left( \text{SInv}(V')[Z^{2^{(h+i)+3}}, T_{h+i+2}] \right)
\]
By Definition 6.2.11, all the edges in a touching path component have the same persistent partition. Hence, by definition and by Lemma 6.4.1, it follows that for \( e = (v_0, v_1) \):
\[
\overline{\text{Reach}}(H, ph)[Z, Z', T] \implies \exists Z^1, \ldots, Z^{2^h} \exists T_0 \geq 0 \ldots \exists T_h \geq 0 \left( T = \sum_{i=0}^{h} T_i \right.
\]
\[
\left( \hat{\rho}(e)[Z, Z^1, T_0] \land \hat{\rho}(e)[Z^{2^h}, Z^1, T_h] \land \right.
\]
\[
\forall 0 \leq T_{h+1} \leq T_0 \exists Z^{2^{h+2}} \left( \hat{\rho}(e)[Z, Z^{2^{h+2}}, T_{h+1}] \land \right.
\]
\[
\text{SInv}(V')[Z^{2^{h+2}}] \land \left( \text{SInv}(V')[Z^{4^{h+1}}] \land \right.
\]
\[
\forall 0 \leq T_{2h+1} \leq T_h \exists Z^{4^{h+1}} \left( \hat{\rho}(e)[Z^{2h}, Z^{4^{h+1}}, T_{2h+1}] \land \right.
\]
\[
\text{SInv}(V')[Z^{4^{h+1}}] \land \left( \text{SInv}(V')[Z^{2^{i+2}}, Z^{2^{i+3}}, T_{i+1}] \land \right.
\]
\[
\forall 0 \leq T_{h+i+2} \leq T_{i+1} \exists Z^{2^{(h+i)+3}} \left( \hat{\rho}(e)[Z^{2^{i+2}}, Z^{2^{(h+i)+3}}, T_{h+i+2}] \land \right.
\]
\[
\text{SInv}(V')[Z^{2^{(h+i)+3}}] \right) \right)
\]
By Definition 6.2.11, the formula \( \hat{\rho}(e) \) is transitive. Hence, we have that if both \( \hat{\rho}(e)[z, z', t_1] \) and \( \hat{\rho}(e)[z'', z, t_2] \) hold, then the formula \( \hat{\rho}(e)[z, z', t_1 + t_2] \) holds. From this consideration, by Lemma 6.4.2, it follows that:
\[
\overline{\text{Reach}}(H, ph)[Z, Z', T] \implies \exists Z^1, \ldots, Z^{2^h} \exists T_0 \geq 0 \ldots \exists T_h \geq 0 \left( T = \sum_{i=0}^{h} T_i \right.
\]
Thus, in particular:

\[
\text{Reach}(H, \rho)[Z, Z', T] \implies \rho(e)[Z, Z', T] \land \\
\forall 0 \leq T' \leq T \exists Z'' (\rho(e)[Z, Z'', T'] \land \text{Inv}([V'])[Z''])
\]

\[6.5\text{ From Reachability to Satisfiability}\]

In this section we consider the problem of limiting the length of the paths we need to consider in the study of reachability. This will lead us to reduce reachability verification to a satisfiability problem.

We first consider the problem of computing a “minimum” elapsed time we need to spend inside a location. In particular, this is always greater than 0 in IDA automata, thanks to condition 2 of Definition 6.2.11.

Since, by Definition 6.2.11, every IDA is in Michael’s form, given two formulæ \(i\) and \(\tau\), the time instants at which a point in \(\text{Sat}(i)\) can reach a point in \(\text{Sat}(\tau)\) with a continuous transition in \(v\) can be characterised by the formula

\[
\eta(v, i, \tau)[T] \overset{\text{def}}{=} \exists Z, Z' (i[Z] \land [Z'] \land \text{Reach}(H, v)[Z, Z', T])
\]

where \(\text{Reach}(v)\) is the formula defined in Section 4.3. On one hand, if this set of time instants is empty, i.e., \(\text{Sat}(i)\) cannot reach \(\text{Sat}(\tau)\), then the following formula is true.

\[
\eta(v, i, \tau) \overset{\text{def}}{=} \forall T \geq 0 \neg \eta(v, i, \tau)[T]
\]

On the other hand, if \(\text{Sat}(i)\) can reach \(\text{Sat}(\tau)\), the infimum of the time instants at which \(\text{Sat}(i)\) reaches \(\text{Sat}(\tau)\) is the solution of the formula:

\[
\mu(v, i, \tau)[T] \overset{\text{def}}{=} \forall \epsilon > 0 \exists T'' (|T' - T| < \epsilon \land \eta(v, i, \tau)[T'']) \land \\
\forall T' \geq 0 (\eta(v, i, \tau)[T'] \implies T \leq T')
\]

We prove now the correctness of the formula \(\mu(v, i, \tau)[T]\). Moreover, we show that if it has a solution this is greater than 0.

**Lemma 6.5.1** Let \(H\) be a hybrid automaton and \(v\) be a location of \(H\). Moreover, let \(i\) and \(\tau\) be two formulæ such that the formula \(\text{MinDyn}(H, v, i, \tau)\) holds, then \(\eta(v, i, \tau)\) does not hold if and only if there exists the infimum, \(t\), of the set \(\{t \mid \eta(v, i, \tau)[t]\}\), \(t > 0\) and \(t\) is the unique value satisfying \(\mu(v, i, \tau)[T]\).
Proof. Let \( \phi \) be the formula \( \phi[Z, Z', T] \equiv \nu[Z] \wedge \tau[Z'] \wedge \text{Dyn}(v)[Z, Z', T] \). By hypothesis, the formula \( \text{MinDyn}(H, v, \iota, \tau) \) holds and thus there exists a \( \hat{t} \geq 0 \) such that, for all \( p \) and \( q \) satisfying \( \iota \) and \( \tau \), respectively, and for all \( t \geq 0 \), if \( \phi[p, q, t] \) holds then \( t \geq \hat{t} \). Hence, since \( \eta(v, \iota, \tau)[T] \) implies the formula \( \exists Z, Z' \phi[Z, Z', T] \) by definition, if \( \eta(v, \iota, \tau)[t] \) holds then \( t \geq \hat{t} > 0 \).

\( (\Rightarrow) \) If the formula \( \overline{\eta}(v, \iota, \tau) \) does not hold, there exists \( t \geq 0 \) such that \( \eta(v, \iota, \tau)[t] \). Thus there exists an infimum \( t \) of the set of \( t \geq 0 \) satisfying the formula \( \eta(v, \iota, \tau)[t] \). Hence for all \( t \geq 0 \), if \( \eta(v, \iota, \tau)[t] \) holds then \( t \geq \hat{t} \). Moreover, for all \( \epsilon > 0 \) there exists a \( t' \) such that \( \|t' - \hat{t}\| < \epsilon \) and \( \eta(v, \iota, \tau)[t'] \) holds by definition of infimum. Thus \( \hat{t} \) satisfies \( \mu(v, \iota, \tau) \). Furthermore, by definition of infimum, \( t' \geq \hat{t} \), and then for all \( \epsilon > 0 \) there exists a \( t' \) such that \( t' < \epsilon + \hat{t} \) and \( t' \geq \hat{t} \). It follows that \( t < \epsilon + \hat{t} \) holds for all \( \epsilon > 0 \). Hence \( \hat{t} \leq t \) and, since \( \hat{t} \) is greater than zero, \( \hat{t} > 0 \).

Now we will prove than exists an unique \( \hat{t} \) satisfying \( \mu(v, \iota, \tau) \). Let \( \tilde{t} \) be such that \( \mu(v, \iota, \tau)[\tilde{t}] \) holds. Thus \( \tilde{t} \) is such that for all \( \epsilon > 0 \) there exists a \( t' \) such that \( \|t' - \tilde{t}\| < \epsilon \) and \( \eta(v, \iota, \tau)[t'] \) and \( \hat{t} \) is smaller or equal than each \( t \) such that \( \eta(v, \iota, \tau)[t] \). Hence \( \hat{t} \) is an infimum for the set of \( t \) satisfying \( \eta(v, \iota, \tau) \). But \( \hat{t} \) is the infimum for such set and then \( \tilde{t} \) is the unique \( \hat{t} \) such that \( \mu(v, \iota, \tau)[\tilde{t}] \).

\( (\Leftarrow) \) Let assume that there exists the infimum, \( \overline{t} \), of the set \( \{t \mid \eta(v, \iota, \tau)[t]\} \), that \( \overline{t} \geq 0 \) and that \( \overline{t} \) is the unique value satisfying \( \mu(v, \iota, \tau)[\overline{t}] \). Thus, there exists a \( \overline{t} > 0 \) such that the formula \( \mu(v, \iota, \tau)[\overline{t}] \) holds. It follows that it does not hold that, for all \( t \geq 0 \), the formula \( \eta(\tau, v, \iota)[t] \) does not hold. Hence \( \overline{\eta}(v, \iota, \tau) \) does not hold by definition.

We are interested in the infimum time we need to spend inside a location \( v \) reached through an edge \( e \) before we can cross another edge \( e' \). The following formula characterises the set of time instants at which the reset of \( e = \langle e', v \rangle \) can reach the activation of \( e' \) subsequent to \( e \).

\[
\xi(e, e')[T] \equiv \eta(v, \text{Reset}(e), \text{Act}(e'))[T]
\]

In Lemma 6.5.2 we will prove that the reset of \( e \) cannot reach the activation of \( e' \) if and only if the following formula holds.

\[
\overline{\xi}(e, e') \equiv \overline{\eta}(v, \text{Reset}(e), \text{Act}(e'))
\]

As a consequence we get that when the reset of \( e \) can reach the activation of \( e' \), the infimum of the instants at which the reset of \( e \) reaches the activation of \( e' \) is the unique solution of the formula

\[
\nu(e, e')[T] \equiv \mu(v, \text{Reset}(e), \text{Act}(e'))[T]
\]

and it is greater than 0. We prove that the above formulae are correct.

Lemma 6.5.2 Let \( H \) be a IDA. If \( e' \in \mathcal{E} \) is subsequence to \( e \in \mathcal{E} \) and \( v \in V \) is the bridge vertex from \( e \) to \( e' \) then \( \mathcal{A}(e') \) is not reachable from \( \mathcal{R}(v) \) with a flow in \( v \) if and only if the formula \( \xi(e, e') \) holds.
Proof. The set \( A(e') \) is not reachable from \( R(e) \) with a flow in \( v \) if and only if \( (v, r) \overset{\lambda} \rightarrow (v, s) \) does not hold for all \( r \in R(e) \) and \( s \in A(e') \). Since, by IDA definition, \( H \) is in Michael’s form, it follows from Theorem 4.3.2 that the set \( A(e') \) is not reachable from \( R(e) \) with a flow in \( v \) if and only if, for all \( r \in R(e) \) and \( s \in A(e') \), there is no \( t \geq 0 \) such that the formula \( \text{Reach}(H, v)[r, s, t] \) holds. Moreover, \( s \in A(e') \) if and only if \( \text{Act}(e')[s] \) by definition and \( r \in R(e) \) if and only if \( \text{Reset}(e)[r] \) by definition. It follows that \( A(e') \) is not reachable from \( R(e) \) with a flow in \( v \) if and only if for all \( t \geq 0 \) it does hold not that there exist \( r \) and \( s \) such that \( \text{Act}(e')[s] \), \( \text{Reset}(e)[r] \), and the formula \( \text{Reach}(H, v)[r, s, t] \) holds. Thus, by definition of the formula \( \xi \), \( A(e') \) is not reachable from \( R(e) \) with a flow in \( v \) if and only if the formula \( \xi(e, e')[T] \) does not hold for any \( t \geq 0 \) and thus, by definition, if and only if \( \bar{\xi}(e, e') \) holds.

At this point we have to characterise the supremum of the time instants at which starting from a set of points we can reach another set of points. On the one hand, when all the variables are reset this supremum is not a real number (i.e., it is \(+\infty\)). In this case we have no problems since reachability can be computed visiting each cycle of the discrete graph at most once. Hence, as already said, we will not consider the case in which there are no independent variables. On the other hand, when some of the variables are not reset their dynamics are constant along a path. Hence we can try to compute the supremum on the projection of all the sets of interest (i.e., initial set, target set and invariants) on the independent variables. If this supremum is a real number, we can use it to bound the length of the traces we have to consider.

The time instants at which a point in the initial set \( \text{Sat}(i) \) can reach a point in the target set \( \text{Sat}(\tau) \) considering only the dynamics of the independent variables are characterised by the formula:

\[
\chi(e, i, \tau)[T] \overset{\text{def}}{=} \exists Z, Z' ((i[Z] \wedge \tau[Z'] \wedge \rho(e)[Z, Z', T]) \wedge \forall 0 \leq T' \leq T \exists Z'' (\rho(e)[Z, Z'', T'] \wedge \text{Inv}(\mathcal{V}'|[Z'']))
\]

where \( \mathcal{T} = (\mathcal{V}', \mathcal{E}') \) is the touching path component of \( e \). When the above formula is not satisfiable the following one is true.

\[
\bar{\chi}(e, i, \tau) \overset{\text{def}}{=} \forall T \geq 0 \neg \chi(e, i, \tau)[T]
\]

Otherwise, the supremum of the time instants which satisfy \( \chi(e, i, \tau)[T] \) is defined by:

\[
\varsigma(e, i, \tau)[T] \overset{\text{def}}{=} \forall \epsilon > 0 \exists T' ((T' - T) < \epsilon \wedge \chi(e, i, \tau)[T']) \wedge \forall T' \geq 0 (\chi(e, i, \tau)[T'] \implies T \geq T')
\]

Lemma 6.5.3 Let \( H \) be a IDA and \( \mathcal{T} = (\mathcal{V}', \mathcal{E}') \) be a touching path component of \( H \). For each pair of edges \( e, e' \in \mathcal{E}' \), the formulæ \( \chi(e, i, \tau)[T] \) and \( \chi(e', i, \tau)[T] \) are equivalent, the formulæ \( \varsigma(e, i, \tau)[T] \) and \( \varsigma(e', i, \tau)[T] \) are equivalent, and the formulæ \( \bar{\chi}(e, i, \tau) \) and \( \bar{\chi}(e', i, \tau) \) are equivalent.
6.5. From Reachability to Satisfiability

Proof. By Definition 6.2.10, $\rho(Dyn(v), I((v, v'))) = \rho(Dyn(v'), I((v, v')))$. Hence, since all the edges in $E'$ have the same persistent partition by Definition 6.2.11, $\rho(Dyn(v), I((v, v'))) = \rho(Dyn(v'), I(v))$ are equivalent for all edges $e \in E'$. Hence, for all pairs of edges $e, e' \in E'$, $\rho(e)[Z, Z', T]$ and $\rho(e')[Z, Z', T]$ are equivalent. It follows that, for all pairs of edges $e, e' \in E'$:

$$\chi(e, t, \tau)[T] \equiv \exists Z, Z'((\epsilon[Z] \land \tau[Z'] \land \rho(e)[Z, Z', T]) \land$$

$$\forall \tau' \leq T \exists Z''(\rho(e)[Z, Z'', T] \land SInv(Z''))$$

$$\equiv \exists Z, Z'((\epsilon[Z] \land \tau[Z'] \land \rho(e')[Z, Z', T]) \land$$

$$\forall \tau' \leq T \exists Z''(\rho(e')[Z, Z'', T] \land SInv(Z''))$$

$$\equiv \chi(e', t, \tau)[T]$$

Furthermore

$$\varsigma(e, t, \tau)[T] \equiv \forall \epsilon > 0 \exists \tau' ((\tau' - T) < \epsilon \land \chi(e, t, \tau)[T']) \land$$

$$\forall \tau' \geq 0 (\chi(e, t, \tau)[T'] \rightarrow T \geq T')$$

$$\equiv \forall \epsilon > 0 \exists \tau' ((\tau' - T) < \epsilon \land \chi(e', t, \tau)[T']) \land$$

$$\forall \tau' \geq 0 (\chi(e', t, \tau)[T'] \rightarrow T \geq T')$$

$$\equiv \varsigma(e', t, \tau)[T]$$

and

$$\overline{\chi}(e, t, \tau) \equiv \forall T \geq 0 \neg \chi(e, t, \tau)[T]$$

$$\equiv \forall T \geq 0 \neg \chi(e', t, \tau)[T]$$

$$\equiv \overline{\chi}(e', t, \tau)$$

As a consequence of Lemma 6.5.3, if $T = \langle \mathcal{V}', \mathcal{E}' \rangle$ is a touching path component of an IDA, we may use the notation $\chi(T, t, \tau)$ to denote the formula $\chi(e, t, \tau)$ where $e$ is any edge in $E'$. Similarly, we will use the notations $\varsigma(T, t, \tau)[t]$ and $\overline{\chi}(T, t, \tau)$ to denote $\varsigma(e, t, \tau)[t]$ and $\overline{\chi}(e, t, \tau)$ for any edge $e$ in $E$.

We prove the correctness of our formulae. In particular, we prove that if the formula $\varsigma(T, t, \tau)[T]$ has a solution this solution is an upper bound for time reachability.

Lemma 6.5.4 Let $H$ be a IDA and $T = \langle \mathcal{V}', \mathcal{E}' \rangle$ be a touching path component of $H$. Moreover, let $t$ and $\tau$ be two formulae. Either the formula $\overline{\chi}(T, t, \tau)$ holds or $t \in \mathbb{R}$ satisfies $\varsigma(T, t, \tau)[T]$ if and only if $\overline{t}$ is the supremum of the set $\{t \mid \chi(T, t, \tau)[t]\}$.

Proof. If the formula $\overline{\chi}(T, t, \tau)$ does not hold, there exists $t_0 \geq 0$ such that the formula $\chi(T, t, \tau)[t]$ holds. Thus the set of $t_0 \geq 0$ satisfying $\chi(T, t, \tau)[t]$ is not empty. Let assume that the supremum $\overline{t}$ of such set exists. Hence for all $t \geq 0$, if $\chi(T, t, \tau)[t]$ holds then $t \geq \overline{t}$. Moreover, for all $\epsilon > 0$ there exists a $t'$ such that $\|t' - \overline{t}\| < \epsilon$ and $\chi(T, t, \tau)[t']$ holds by definition of supremum. Thus $\overline{t}$ satisfies $\varsigma(T, t, \tau)$.
Now we will prove that if \( \varsigma(\mathcal{T}, t, \tau)[\bar{t}] \) holds, then \( \bar{t} \) is the supremum of the set \( \{ t \mid \chi(\mathcal{T}, t, \tau)[t] \} \). Let \( \bar{t} \) be such that \( \chi(\mathcal{T}, \bar{t}, \tau)[\bar{t}] \) holds. Thus \( \bar{t} \) is such that for all \( \epsilon > 0 \) there exists a \( t' \) such that \( ||t' - \bar{t}|| < \epsilon \) and \( \chi(\mathcal{T}, t, \tau)[t'] \) and \( \bar{t} \) is greater or equal than each \( t \) such that \( \chi(\mathcal{T}, t, \tau)[t] \). Hence \( \bar{t} \) is the supremum for the set of \( t \) satisfying \( \chi(\mathcal{T}, t, \tau) \).

To conclude, we need to introduce a formula \( \theta(\mathcal{T}, j, t, \tau) \) which expresses the fact that either the set represented by \( t \) cannot reach the set represented by \( \tau \) with respect to the locations and edges in \( \mathcal{T} \) or it can be reached with less than \( j + 1 \) discrete transitions. Such formula is defined as:

\[
\theta(\mathcal{T}, j, t, \tau) \equiv \chi(\mathcal{T}, t, \tau) \lor \left( \bigwedge_{(e, e') \in \text{Sb}(\mathcal{T})} \xi(e, e') \lor \exists T \left( \chi(\mathcal{T}, t, \tau)[T] \land \left( \bigwedge_{(e, e') \in \text{Sb}(\mathcal{T})} \forall T' (\nu(e, e')[T'] \rightarrow j \star T' > T) \right) \right) \right)
\]

where \( \text{Sb}(\mathcal{T}) \) denotes the set of pairs \( (e, e') \) such that \( e' \) is subsequent to \( e \) in \( \mathcal{T} \). The following theorem proves the correctness of \( \theta(\mathcal{T}, j, t, \tau) \).

**Theorem 6.5.5** Let \( H \) be a IDA and \( \mathcal{T} \) be a touching path component of \( H \). Moreover, let \( t \) and \( \tau \) be two formulæ. If the set \( \{ t \mid \chi(\mathcal{T}, t, \tau)[t] \} \) has a supremum in \( \mathbb{R} \), there exists a \( j \in \mathbb{N} \) such that \( \theta(\mathcal{T}, j, t, \tau) \) holds. Furthermore, if \( \theta(\mathcal{T}, j, t, \tau) \) holds and there exist \( p \) and \( q \) such that \( \iota[p] \) and \( \tau[q] \) and \( q \) is reachable from \( p \) in \( H \), then \( q \) is reachable from \( p \) in \( H \) with less than \( j + 1 \) discrete transitions.

**Proof.** By definition, \( \theta(\mathcal{T}, j, t, \tau) \) holds if and only if one of the followings hold:

- \( \chi(\mathcal{T}, t, \tau) \)
- \( \left( \bigwedge_{(e, e') \in \text{Sb}(\mathcal{T})} \xi(e, e') \right) \)
- \( \exists T \left( \chi(\mathcal{T}, t, \tau)[T] \land \left( \bigwedge_{(e, e') \in \text{Sb}(\mathcal{T})} \forall T' (\nu(e, e')[T'] \rightarrow j \star T' > T) \right) \right) \)

If either \( \chi(\mathcal{T}, t, \tau) \) holds or \( \left( \bigwedge_{(e, e') \in E'} \xi(e, e') \right) \) holds then, the formula \( \theta(\mathcal{T}, 0, t, \tau) \) holds. Let us assume that both the formulæ \( \chi(\mathcal{T}, t, \tau) \) and \( \left( \bigwedge_{(e, e') \in E'} \xi(e, e') \right) \) do not hold.

If the formula \( \left( \bigwedge_{(e, e') \in E'} \xi(e, e') \right) \) does not hold, then there exist pairs of edges in \( E' \), \( e = (v', v) \) and \( e' = (v, v'') \), such that \( \xi(e, e') \) does not hold. Hence, by the definition of the formulæ \( \mathcal{P} \) and by Lemma 6.5.1, for each of these pair, \( e \) and \( e' \), there exists a unique \( t \in \mathbb{R} \) satisfying \( \nu(e, e')[T] \) and such \( t \) is greater than zero. Let \( \bar{t} \) be the minimum \( t \), over all these pairs, satisfying \( \nu(e, e')[T] \). Moreover, by hypothesis, there exists the supremum, \( \bar{t} \), of the set \( \{ t \mid \chi(\mathcal{T}, t, \tau)[t] \} \). Thus, since \( \chi(\mathcal{T}, t, \tau) \) does not hold, \( \bar{t} \) is the unique satisfies \( \varsigma(\mathcal{T}, t, \tau)[T] \) by Lemma 6.5.4. Hence \( \bar{j} = [\bar{t}/t] + 1 \) is
a natural number. Furthermore, for any pair \((e, e') \in Sb(J)\), if \(\nu(e, e')[t]\) then, \(t \geq \bar{t}\) and thus:
\[
\bar{t} + t + 1 \geq \bar{t} + 1 + 1 > \bar{t}
\]
Hence \(\bigwedge_{e, e' \in Sb(J)} \forall T (\nu(e, e')[T] \rightarrow \bar{t} + T > \bar{t})\) holds. It follows that \(\theta(J, \bar{t}, \tau)\) holds and hence if the set \(\{t \mid \chi(J, \bar{t}, \tau)[t]\}\) has a supremum, there exists \(\bar{t} \in \mathbb{N}\) such that \(\theta(J, \bar{t}, \tau)\) holds.

Now we will show that if \(\theta(J, \bar{t}, \tau)\) holds, there exist \(p\) and \(q\) such that \(\nu[p]\) and \(\tau[q]\) and \(q\) is reachable from \(p\) in \(H\) then, \(q\) is reachable from \(p\) in \(H\) with less than \(\bar{t} + 1\) resets.

Let us assume that \(\chi(J, \bar{t}, \tau)\) holds. Hence \(\theta(J, 0, \bar{t}, \tau)\) holds too. By definitions of the formulae \(\chi(J, \bar{t}, \tau)\) and \(\chi(J, \bar{t}, \tau)\):
\[
\chi(J, \bar{t}, \tau) \equiv \forall T \geq 0 \neg \chi(J, \bar{t}, \tau)[T]
\]
\[
\equiv \forall T \geq 0 \neg (\exists Z, Z' \ (\nu[Z] \land \tau[Z'] \land \rho(e)[Z, Z', T] \land \\
\forall 0 \leq T' \leq T \exists Z' \ (\rho(e)[Z, Z', T'] \land SInv(V[Z'])))
\]
\[
\equiv \forall T \geq 0 \forall Z, Z' \ (\nu[Z] \lor \neg \tau[Z'] \lor \neg \rho(e)[Z, Z', T] \lor \\
\neg (\forall 0 \leq T' \leq T \exists Z' \ (\rho(e)[Z, Z', T'] \land SInv(V[Z']))))
\]
Thus, by Lemma 6.4.3, it holds that for any path \(ph \in J\):
\[
\chi(J, \bar{t}, \tau) \implies \forall T \geq 0 \forall Z, Z' \ (\nu[Z] \lor \neg \tau[Z'] \lor \neg \text{Reach}(H, ph)[Z, Z', T])
\]
By Lemma 4.3.3 and Lemma 6.1.7, it follows that there are no \(p\) and \(q\) such that \(\nu[p]\) and \(\tau[q]\) holds. Hence \(\theta(J, 0, \bar{t}, \tau)\) holds too. Moreover, by Lemma 6.5.2, for all edges pair \((e, e') \in E'\) for all vertices \(v \in V'\) such that \(e'\) is subsequent to \(e\) and \(v\) is the bridge vertex from \(e\) to \(e'\), \(A(e')\) is not reachable from \(R(e)\) with a flow in \(v\). Furthermore, it may exist an edge \((v, v') \in E'\) and a \(p \in \mathbb{R}^k\) such that \(\nu[p]\) and \(p\) reaches \(A(v, v')\). Thus for all \(p\) and \(q\) such that \(\nu[p]\) and \(\tau[q]\) holds, \(q\) is reachable from \(p\) in \(J\) with at most one reset.

Let assume that neither \(\chi(J, \bar{t}, \tau)\) nor \(\chi(J, \bar{t}, \tau)\) hold. As proved above, it follows that there exists \(t\) such that \(\chi(J, \bar{t}, \tau)[\bar{t}]\). Moreover, for each \((e, e') \in Sb(J)\) it holds that either \(\nu(e, e')[t]\) has not a solution, since it is not possible to reach the activation of \(e'\) from the reset of \(e\), or \(\nu(e, e')[T]\) is satisfied by \(\nu(e', e') \in \mathbb{R}\) such that \((\bar{t} + \bar{t}, \bar{t}) > \bar{t})\) holds. Moreover, by Lemma 6.5.4, \(\bar{t} \in \mathbb{R}\) satisfies \(\chi(J, \bar{t}, \tau)[T]\) if and only if \(\bar{t}\) is the supremum of the set \(\{t \mid \chi(J, \bar{t}, \tau)[t]\}\). Thus, for all \(t > \bar{t}\), the formula \(\chi(J, \bar{t}, \tau)[t]\) does not hold. Hence, by Lemma 6.4.3, by Lemma 4.3.3 and by the definition of the formula \(\chi\), if there exist \(p\) and \(q\) such that \(\nu[p]\) holds, \(\tau[q]\) holds and \(q\) is reachable from \(p\) in \(H\) with time \(t\), then \(t \leq \bar{t}\). Moreover, by Lemma 6.5.1, \((e, e') \in \mathbb{R}\) satisfies \(\nu(e, e')[T]\) if and only if \(\nu(e, e')\) is the infimum of the set \(\{t \mid \eta(v, \text{Reset}(e), Act(e'))\}\), where \(v\) is the bridge vertex from \(e\) to \(e'\). Thus for all \(t < \bar{t} (e, e')\), the formula \(\eta(v, \bar{t} (e, e'), Act(e'))[t]\) does not hold. Hence, by Flow’s definition and by Lemma 4.3.3,
if there exist \( p', q' \) and \( t' \) such that \( \text{Act}(e')[q'] \) holds, \( \text{Reset}(e)[p'] \) holds and \( p' \) reaches \( q' \) in \( v \) in time \( t' \), then \( t' \geq \bar{t}(e,e') \). Furthermore, \( \text{Act}(e')[q'] \) holds if and only if \( q' \in \mathcal{A}(e') \) and \( \text{Reset}(e)[p'] \) if and only if \( p' \in \mathcal{R}(e) \). Thus for each pair of edges \( e \) and \( e' \) in \( \mathcal{T} \) and for all vertices \( v \) such that \( v \) is the bridge vertex from \( e \) in \( e' \), if there exist \( p', q' \) and \( t' \) such that \( p' \in \mathcal{R}(e) \), \( q' \in \mathcal{A}(e') \) and \( p' \) reaches \( q' \) in \( v \) in time \( t' \), then \( t' \geq \bar{t}(e,e') \). Hence if there exist \( p, q \) such that \( \epsilon[p] \) holds, \( \tau[q] \) holds and \( p \) reaches \( q \) through a trace \( tr \) in \( H \), whose corresponding path is \( ph = v_0, \ldots, v_n \), the automaton stays in each location \( v_i \) of \( tr \) at least for time \( \bar{t}((v_{i-1}, v_i), (v_i, v_{i+1})) \). Let \( t \) be the minimum of the infini \( \bar{t}(e,e') \) with \( e \) and \( e' \) subsequent in \( \mathcal{T} \). Thus the number of resets in the trace \( tr \) must be less than \( \lceil t/\bar{t} \rceil + 1 \). Since \( l \) is one of the infini, we have that \( j \geq \bar{t} \), and hence \( j + 1 > \lceil t/\bar{t} \rceil + 1 \). It follows that if there exist \( p, q \) such that \( \epsilon[p] \) holds, \( \tau[q] \) holds and \( p \) reaches \( q \) through a trace \( tr \) in \( H \), the number of resets in the trace \( tr \) must be less than \( j + 1 \). \hfill \( \blacksquare \)

Exploiting Theorem 6.5.5 and Lemma 4.3.3 and assuming the decidability of the IDA’s theory, we can write Algorithm 4 to decide whether there exist \( p \) and \( q \) such that \( \epsilon[p] \) holds, \( \tau[q] \) holds and \( p \) reaches \( q \) in an IDA \( H \). The algorithm works when the involved suprema are real numbers. The following formula is used to exit the repeat loop:

\[
\text{Test}(ph, \iota, \tau) \overset{\text{def}}{=} \exists T \geq 0 \exists Z, Z' \left( \iota[Z] \land \tau[Z'] \land \text{Reach}(H, ph)[Z, Z', T] \right)
\]

**Algorithm 4** Check whether there exist \( p \) and \( q \) such that \( \epsilon[p] \) holds, \( \tau[q] \) holds and \( p \) reaches \( q \) in a IDA \( H \)

**Require:** An IDA \( H \) and two formulæ \( \iota \) and \( \tau \) such that the formula \( \exists T \geq 0 \lhd (\mathcal{E}, \iota, \tau)[T] \) holds.

**Ensure:** Return TRUE if there exist \( p \) and \( q \), FALSE otherwise.

for all \( \mathcal{T} \) touching path component in \( H \) do

\( j \leftarrow 0 \)

repeat

for all \( ph \) path of length \( j \) in \( \mathcal{T} \) do

if \( \text{Test}(ph, \iota, \tau) \) then

Return TRUE

end if

end for

\( j \leftarrow j + 1 \)

until \( \theta(\mathcal{T}, j - 1, \iota, \tau) \)

end for

Return FALSE

The termination of the presented algorithm is ensured by the first part of Theorem 6.5.5, while the correctness follows by Lemma 4.3.3 and by the second part of Theorem 6.5.5.
One could argue that the standard reachability algorithm suggested by Proposition 3.2.10 can be applied instead of Algorithm 4. However, if there exists a trajectory from \( \iota \) whose corresponding path is infinite and \( \tau \) is not reachable from \( \iota \), then the former technique does not terminate since it discovers at each iteration new points belonging to the reachable set. On the other hand, Algorithm 4 performs reachability considering traces of bounded length only.

**Corollary 6.5.6** Let \( H \) be an IDA and \( \iota \) and \( \tau \) be two formulæ. If the set \( I_H^{\nu,p} \) is bounded for all \( \nu \in \mathcal{V} \) and all \( p \in \mathbb{R}^k \), then the Algorithm 4 can be used to decide whether in \( H \) \( \iota \) can reach \( \tau \).

**Proof.** The correctness of our algorithm follows by the second part of Theorem 6.5.5 and by Lemma 4.3.3. Termination is ensured by the first part of Theorem 6.5.5. \( \blacksquare \)

### 6.6 IDA and Bisimulation

In this section we prove that there exists an IDA which does not admit a finite bisimulation quotient. In particular, we show that the FOCoRe automaton \( H_{inf} \) presented in Section 5.3, is an IDA also.

**Lemma 6.6.1** The automaton \( H_{inf} \), defined in Section 5.3, is an IDA.

**Proof.** We prove each condition of Definition 6.2.11.

**Condition 1** Since \( H_{inf} \) is a FOCoRe by Lemma 5.3.1, \( H_{inf} \) is in Michael’s form.

**Condition 2** There is only one pair of subsequent edges in \( H_{inf} \): \((e, e)\). By \( \text{Dyn} \)’s definition, if \( \text{Dyn}(\nu)[Z, Z', T] \) then \( |Z - Z'| \leq T \). Moreover, it is easy to see that the minimum value for \( |x - x'| \) when \( x \in \mathcal{R}(e) \) and \( x' \in \mathcal{A}(e) \) is 2. Hence, if \( \text{Reset}(e)[Z] \land \text{Act}(e)[Z'] \land \text{Dyn}(\nu)[Z, Z', T] \), then \( T \geq 2 \). It follows that the formula \( \text{MinDyn}(H, \nu, \text{Reset}(e), \text{Act}(e')) \) holds for all couples \((e, e')\) of subsequent edges in \( H_{inf} \).

**Condition 3** Consider the partition \( \mathcal{P} = \{I = \emptyset, \mathcal{D} = \{Z_1, Z_2\}\} \).

- \( I \) is obviously independent on both \( e \) and \( \nu \);
- \( \rho(\text{Dyn}(\nu), I) \equiv tt \equiv \rho(\text{Dyn}(\nu), I) \);
- \( \rho(\text{Reset}(e), I) \equiv tt \equiv \bigwedge_{Z_i \in \emptyset} Z_i = Z_i' \).

Hence \( \mathcal{P} \) is a persistent partition of \( e \) and indeed it is a persistent partition for all edges of \( H_{inf} \).

**Condition 4** Since \( \rho(\text{Dyn}(\nu), I) \equiv tt, \rho(\text{Dyn}(\nu), I)[Z, Z', T + T'] \) holds for all \( Z, Z', T \) and \( T' \). It follows that \( \rho(\text{Dyn}(\nu), I) \) is transitive for each location \( \nu \) of \( H_{inf} \).

Indeed all the conditions in IDA’s definition are satisfied by \( H_{inf} \) and then \( H_{inf} \) is an IDA. \( \blacksquare \)
Theorem 6.6.2  There exist IDA having infinite bisimulation quotient.

Proof. The proof follows directly from Theorem 5.3.5 and Lemma 6.6.1.  ■
III

Approximating Analysis
“The Devil is in the Details”

Unknown

As reported in the first part of this thesis, checking safety properties on hybrid automata can be reduced to the reachability problem; in particular, to prove that a safety property $\varphi$ is always true for a hybrid automaton $H$, we only need to prove that all the states in which $\varphi$ is false are not reachable from a set of initial states of $H$. Since, it has been proved in [82] that, in general, the reachable set is not computable, many works propose over-approximations techniques to estimate such set. In [74] Halbwachs et al. suggested convex approximations as a way to verify linear hybrid systems, Dang and Maler proposed to verify hybrid automaton properties via face lifting in [54], Chutinan and Krogh showed in [45] how evolutions of polyhedral-invariant hybrid automata can be approximated using polyhedra, Asarin et al. gave in [12] a technique to approximate reachability analysis of piecewise-linear dynamical systems, Kurzhanski and Varaiya introduced ellipsoidal techniques in [88, 89], Botchkarev and Tripakis showed how to extend ellipsoidal techniques to threat linear differential inclusion dynamics in [28], and Alur et al. proposed in [5, 6, 7] predicate abstraction as a technique to perform reachability analysis. In the last years, many other approaches have been introduced. They are mainly based on either Lyapunov functions [85, 31], cylindrical algebraic decomposition [131], viability theory [58, 14], or Hamilton-Jacobi equation [111, 110].

All the above approximating techniques face two orthogonal problems. The first one is an algorithmic problem and it is about how the approximation is computed. The second concerns the way in which regions are represented and it involves the data structures used during computations. Even if many of the above techniques relate a specific representation to a particular algorithm, in general algorithms do not constrain the representation method.

This chapter describes a work on reachability analysis algorithms of linear hybrid systems. The starting point is an algorithm for over-approximation of reachability
regions due to Botchkarev [27]. In its original definition, such algorithm represents region by ellipsoids and it approximates the flow-tube of reached sets by the union of partial flow-tubes obtained by discretising the integration time. We present a revised reachability computation that avoids the approximations caused by the union operation in the discretised flow-tube estimation. The new algorithm may correctly classify as unreachable, states that the previous one certifies as reachable due to looser over-approximations. Moreover, we give theoretical results on convergence of both algorithms, introducing a class of linear hybrid automata on which they terminate. Notice that such results are independent from the region representation technique and they concern only the way in which computations are preformed.

We recall the basic notation used in this chapter and introduced in Chapter 3. Given a hybrid automaton $H$, a region $R$ is a set of states such that all the states are in same location. The region $R = \{ (v, r) \mid \varphi[r] \}$ of the states in location $v$ whose continuous value satisfies $\varphi$ is denoted by $\langle v, \varphi \rangle$. We use $\langle R \rangle_{t}^-$ and $\langle R \rangle_{t}^-$ to indicate the set of states reachable from $R$ with a flow of time $t' \leq t \in \mathbb{R}_{\geq 0}$ and $\bigcup_{t \in \mathbb{R}_{\geq 0}} \langle R \rangle_{t}^-$, respectively (i.e., $\langle R \rangle_{t}^- \overset{\text{def}}{=} \{ q \mid \exists Z \exists T \geq 0 \langle v, Z \rangle \rightarrow_{C} q \land \varphi[Z] \}$) and $\langle R \rangle \overset{\text{def}}{=} \{ q \mid \exists Z \exists T \geq 0 \langle v, Z \rangle \rightarrow_{C} q \land \varphi[Z] \}$). Moreover, we use $\langle R \rangle_e^-$ to denote the set of states reachable from $R$ with discrete transition on an $H$’s edge $e \in \mathcal{E}$ i.e., $\langle R \rangle_e^- \overset{\text{def}}{=} \{ \langle v', Z' \rangle \mid \exists Z, Z' \phi[Z] \land \langle v', Z \rangle \rightarrow_{D} (v'', Z') \land v = v' \}$. Finally, we write $\langle R \rangle_{0}^-$ meaning $\text{Sat}(\varphi)$. Notice that if $R$ is a region, then $\langle R \rangle_i^-, \langle R \rangle^-$, and $\langle R \rangle_e^-$ are regions also.

### 7.1 Botchkarev’s Algorithm

A technique used to represent and operate on sets in reachability analysis is sometimes called a calculus method. Three calculi commonly used are polyhedral calculus [79, 10], ellipsoidal calculus [88, 27], and zonotopic calculus, recently introduced in hybrid automaton context by Girard [73, 68]. These calculi represent sets using polyhedra, ellipsoids, and zonotope, respectively.

An algorithm based on ellipsoidal representations was presented by Botchkarev in [27]. The algorithm over-approximates the reachable set of a linear hybrid automaton and in fact can be used with any calculus method that represents sets using compact and convex regions, computes the union, intersection and geometric sum of regions and evaluates the linear transformations of regions. If Botchkarev’s algorithm returns UNREACHABLE on a given linear hybrid automaton $H$, two regions of states $R_I$ and $R_T$, and a real number $t$, then the over-approximation guarantees that $H$ cannot reach $R_T$ from $R_I$ in time $t$.

To evaluate the reach set, the algorithm maintains a table $\mathcal{T}$ of pairs. Each pair has the form $\langle R, t_R \rangle$, where $R$ is a region of states and $t_R$ is a positive real number. When $\langle R', t_{R'} \rangle$ is in $\mathcal{T}$, the region $R'$ is considered as reachable from $R_I$ by automaton $H$ in time $t_{R'}$. At the beginning of the computation, $\mathcal{T}$ contains the pair $\langle R_0, t_0 \rangle$ only, where $R_0$ is the set of initial states $R_I$ and $t_0$ is equal to $0$. The algorithm, essentially, iterates three steps:
1. pick a pair \( \langle R, t_R \rangle \) from \( T \);

2. find states reachable from \( R \) computing an over-approximation of the flow-tube from \( R \) and, for each arc \( e \), calculate:
   - the intersection, \( D_e \), of the flow-tube with the activation of \( e \);
   - the set \( [D_e]_e^- \) of states reachable from \( D_e \) traversing arc \( e \);  
3. insert into \( T \) a new pair, \( \langle [D_e]_e^-, t_R + t_{new} \rangle \), where \( t_{new} \) is an under-estimation of the flow time needed to reach \( A(e) \) from \( R \).

The key point in Botchkarev’s approach is the discretisation of time, by a time step, \( \delta \). Given the parameter \( \delta \), for each \( k \in \{0, \ldots, \left\lfloor \frac{t_{final}}{\delta} \right\rfloor \} \), the algorithm starts by computing the set, \( R_k \), of points reachable from \( R \) at time \( t = k\delta \). Since some of the states that are reachable with a \( t' \)-timed flow, where \( k\delta \leq t' \leq (k + 1)\delta \), \( t > 0 \), may not be included in these approximations, the algorithm expands each set \( R_k \) by a ball \( B_t(\vec{0}) \), centred in \( \vec{0} \) and with radius \( \epsilon \). The radius \( \epsilon \) is set in such a way to account for every state reachable at time \( t \leq t' \leq t + \delta \). Lemma 7.1.1 (from [27]) shows how the radius \( \epsilon \) can be evaluated beforehand as a function of \( \delta \) and of the flow function for the current location.

**Lemma 7.1.1** Let \( R = \langle v, \varphi \rangle \) be a region of states and \( \dot{Z} = A_vZ + u \) be the differential equation defining the dynamics in \( v \), where \( A_v \) is an \( n \times n \)-matrix and \( u \) is vector belonging to a compact and convex set \( U_v \subseteq \mathbb{R}^n \). The following estimate is true for all \( t \in [0, \delta] \):

\[
[R]_t^+ \subseteq [R]_0 \oplus B_{e_v, \delta}(\vec{0})
\]

where \( \oplus \) is the Minkowski sum and

\[
\epsilon_{v, \delta} = (e^{N_{A_v} \delta} - 1)D + e^{N_{A_v} \delta}N_{U_v} \delta,
\]

with \( N_{A_v} = \|A_v\| \), \( N_{U_v} = \max_{u \in U_v} \|u\| \) and \( D = \max_{x \in [R]_k} \|x\| \).

**Proof.** See the proof in [27].

From now on, we will denote with \( T_{\delta,k}(R) \) the function that evaluates the region reached with a \( k\delta \)-timed flow from the region \( R \) i.e., \( T_{\delta,k}(R) \overset{def}{=} \{ \ell' \mid \exists \ell \in R \ \ell \xrightarrow{\delta\ell} C \ \ell' \} \). Furthermore, given a region \( R = \langle v, \varphi \rangle \) and an edge \( e = \langle v, v' \rangle \), the notation \( V_{\delta}^e \) and \( W_{\delta}^e \) will denote an over-approximations of the intersection between the flow-tube from \( R \) and the activation of the edge \( e \) (i.e., \( V_{\delta}^e(R) \supseteq \left[\left[ (R)_{c}^- \right]_0 \cap A(e) \right] \) and the reset of \( (R)_{c}^- \) over \( e \) (i.e., \( W_{\delta}^e(R) \supseteq \left[\left[ (R)_{c}^- \right]_0 \right]_1 \)), respectively.

The parameters of Botchkarev’s algorithm are: an automaton \( H \), an initial region \( R_I = \langle v_I, \varphi_I \rangle \), a state region \( R_T = \langle v_T, \varphi_T \rangle \) whose reachability must be tested, the time horizon \( \ell \), and a time step \( \delta \). During the initialisation phase the pair \( \langle R_I, 0 \rangle \) is inserted into \( T \). The steps of the algorithm can be described as follows:
1. if $\mathcal{T}$ is empty then stop and return UNREACHABLE. Otherwise, pick in $\mathcal{T}$ a pair $\langle R, t_R \rangle$, with the smallest $t_R$ and where $R = \langle v, \varphi \rangle$, and remove it from the table;

2. for the selected pair $\langle R, t_R \rangle$:
   
   (a) evaluate $S_{v,k}^+ = ([T_{0,k} \circ (R)]_{\delta} + B_{v,k}(\bar{0})) \cap \mathcal{I}(v)$, for $k \in \{0, \ldots, m\}$, where $m$ is the minimum between $\lceil \bar{t} - t_R \delta \rceil$ and the smallest $k$ such that $S_{v,k}^+ \cap \mathcal{I}(v) = \emptyset$;
   
   (b) if $v = v_T$ and $S_{v,k}^+ \cap [R_T]_{\delta} \neq \emptyset$; for some $k \in \{0, \ldots, m\}$, then return REACHABLE;
   
   (c) let $E = \{e_1, \ldots, e_l\}$ be the set of edges leaving location $v$ i.e., $e_i = \langle v, v_i \rangle$ for all $i \in [1, l]$.

   i. For each $e \in E$, set the regions $V_{e_i}^+$ and $W_{e_i}^+$ to $\emptyset$;
   
   ii. Let $j_{\min(e_i)}$ be the minimum value between $m + 1$ and the smallest $k$ such that $S_{v,k}^+ \cap \mathcal{A}(e_i) \neq \emptyset$. If $0 \leq j_{\min(e_i)} \leq m$ then compute
   $$ V_{e_i}^+ \supseteq \bigcup_{j = j_{\min(e_i)}}^{m} (S_{v,j}^+ \cap \mathcal{A}(e_i)) . $$
   
   iii. If $V_{e_i}^+ \neq \emptyset$ then compute:
   $$ W_{e_i}^+ \supseteq \left( \left( V_{e_i}^+ \right) \right)_{e_i}^- . $$

3. let $\tau_i$ be such that $\tau_i = j_{\min(e_i)} \delta$. Then, for each $W_{e_i}^+ \neq \emptyset$ computed at step 2(c)iii, add to $\mathcal{T}$ the pair $\langle R_i, t_R + \tau_i \rangle$, where $R_i = \langle v_i, W_{e_i}^+ \rangle$, and repeat from step 1.

Termination of Botchkarev’s algorithm is not guaranteed. As a matter of fact, if we apply this algorithm to a Zeno automaton, the table $\mathcal{T}$ will not necessarily become empty.

Note also that, because of the over-approximation of the flow-tube, the algorithm may fail to terminate even on a non-Zeno automaton.

**Theorem 7.1.2 (Non-Termination Result)** There exist a non-Zeno hybrid automaton $H$ and an initial region $\langle v_S, S \rangle$ such that for all time steps $\delta$ the Botchkarev’s algorithm fails to terminate.

**Proof.** Consider the hybrid automaton presented in Figure 7.1. The flow starting from $\langle v, S \rangle$, where $S = \{(1, 0)\}$, does not enable any arc. As a matter of fact, the flow starting from $\langle v, S \rangle$ is computable and equal to the region $\langle v, \bar{S} \rangle$, where $\bar{S} = \{(Z_1, Z_2) \mid Z_1 \geq 1 \land Z_2 = 0\}$. Thus the automaton of Figure 7.1 is not a Zeno automaton. Nevertheless, the differential equation associated to the flow in location $v$ is $\dot{Z} = AZ + u$, where $Z = (Z_1, Z_2)$, $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $u = (0, 0)$. Thus, by
Lemma 7.1.1. If \( N_A > 0 \), \( N_U = 0 \) and \( D > 0 \) and then the ball \( B_{\epsilon_v,\delta}(\vec{0}) \) used in step 2a of the algorithm has radius \( \epsilon_v,\delta > 0 \). Indeed, \( S_{\epsilon,0}^+ \) intersects the arc’s guard and this does not depend on the choice of the calculus method. Thus, the pair \((\bar{R},0)\), where \( \bar{R} = (v,S) \), will be inserted into \( T \), bringing back the algorithm to the initial state.

Although termination is not guaranteed in general, there are restricted classes of hybrid automata for which termination can be proved. In particular, in the next section, we introduce a class of hybrid automata for which we are able to show a finite upper bound on the number of steps of Botchkarev’s algorithm.

### 7.2 Some Theoretical Results on Termination

As shown in the previous section, to guarantee convergence of the reachability algorithm, it is not enough to consider only non-Zeno automata, because the over-approximations made in step 2a may lead the algorithm to go on forever even with a non-Zeno automaton, as shown by the example of Figure 7.1. For this reason, we would like to restrict non-Zeno automata to identify a class such that the expansions made by the algorithm at step 2a and the approximations produced by calculus method do not cause an infinite sequence of arc activations. To that purpose we introduce the definitions of \( \epsilon_H \)-expansive hybrid automaton, approximation index of a calculus method and \((C, \delta)\)-compatibility. The idea is that, if \( e \) is an arc and \( R \) is a state region, if a hybrid automaton is \( \epsilon_H \)-expansive, then expanding \([R]_e^-\) by means of an \( \epsilon_H \)-ball, no arc at the destination location is enabled.

**Definition 7.2.1 (\( \epsilon_H \)-Expansive Hybrid Automaton)** A hybrid automaton \( H \) is \( \epsilon_H \)-expansive, where \( \epsilon_H \in \mathbb{R}_{\geq 0} \) is the expansion radius, if, for each pair of subsequent arcs \( e_1 = (v_1, v_2), e_2 = (v_2, v_3) \in \mathcal{E} \):

\[
\left( R(e_1) \oplus B_{\epsilon_H}(\vec{0}) \right) \cap A(e_2) = \emptyset
\]

Notice that \( \epsilon_H \)-expansiveness does not imply non-Zenoness. As a matter of fact, the following result holds.

**Theorem 7.2.2** There exist Zeno automata which are \( \epsilon_H \)-expansive, with \( \epsilon_H > 0 \).
Proof. Consider the hybrid automaton \( H \) such that locations are \( \{v\} \), edges are \( \{(v, v')\} \), \( Inv(v)[Z] \overset{\text{def}}{=} 0 \leq Z_1 \land Z_1 \leq 1 \), \( Act(e)[Z] \overset{\text{def}}{=} Z_1 = 1 \), \( Reset(e)[Z, Z'] \overset{\text{def}}{=} Z_2 = 2 * Z_2 \land Z_1 = 0 \), and \( Dyn(v)[Z, Z', T] = Z_1' = Z_2 \land T + Z_1 \land Z_2 = Z_2 \). There is only a pair of subsequent edges in \( H \): \( e \) and itself. Obviously \( (R(e) \oplus B_{e_H}(\emptyset)) \cap A(e) = \emptyset \) holds for all \( \epsilon_H \in [0, 1) \), hence, \( H \) is \( \epsilon_H \)-expansive for all \( \epsilon_H \in [0, 1) \). However, \( H \) is Zeno. As a matter of fact, let us consider the trajectory from the state \( \langle v, (0, 1) \rangle \). From dynamics definition, it follows that \( H \) elapses time \( t = \frac{n - 1}{r_2} \) to reach a state \( q_S = \langle v, (s_1, r_2) \rangle \) from \( \langle v, (r_1, r_2) \rangle \). Moreover, the reset is activated for any state \( \langle v, (1, r_2) \rangle \) and the reset brings \( H \) from a generic state \( \langle v, (1, r_2) \rangle \) to a state of the type \( \langle v, (0, 2 * r_2) \rangle \). Thus the total amount of time, \( t_S \), elapsed by a \( H \)'s trajectory from \( q_S \) is:

\[
t_S = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{2^n} + \ldots = \sum_{i=0}^{\infty} \frac{1}{2^i}
\]

\[
< \frac{1}{1 - \frac{1}{2}} = 2
\]

It follows that \( H \) has a Zeno trajectory and it is Zeno.

\( \blacksquare \)

As we said, to obtain termination, it is not enough to guarantee that the automaton has no infinite step-based run. We also need to guarantee that the approximation made by the calculus method does not incorrectly enable any arc. For this reason, we introduce the approximation index of a calculus method. Given an arc \( e = \langle v, v' \rangle \) and a calculus method \( C \), let \( V^e_C \) be the best over-approximation of the region \( V_e = (v, Act(e)) \) in \( C \).

Definition 7.2.3 (Approximation Index) The approximation index, \( \gamma_H^C \), of calculus method \( C \) on hybrid automaton \( H \) is:

\[
\gamma_H^C = \max_{e \in E} h_+ \left( \left[ [V^e_C]_{\epsilon_H} \right]_{\epsilon_H}, R(e) \right),
\]

where \( h_+ \) is the Hausdorff’s semi-distance between two sets of states.

Definition 7.2.4 ((\( C, \delta \))-Compatibility) Let \( H \) be an \( \epsilon_H \)-expansive hybrid automaton and \( \dot{Z} = A_v Z + u \), with \( u \in U_v \), be the differential equation defining the dynamics in \( v \). Suppose that \( \gamma_H^C \) is the approximation index of calculus method \( C \) on \( H \). If there exists a \( \delta \in \mathbb{R}_{\geq 0} \) such that, for each edge \( e = \langle v, v' \rangle \),

\[
\epsilon_H - \gamma_H^C > \left( \epsilon_N^{A_v} - \delta \right) D(e) + \epsilon_N^{U_v} N_{U_v} \delta,
\]

where \( D(e) = \max_{s \in [V^e_C]_{\epsilon_H}} \| s \| \), \( N_{A_v} = \| A_v \| \) and \( N_{U_v} = \max_{u \in U_v} \| u \| \), then, we say that \( H \) is compatible with calculus method \( C \) and time step \( \delta \), i.e., that \( H \) is \((C, \delta)-\)compatible.

The next theorem gives an upper bound on the number of cycles required by Botchkarev’s algorithm when it runs on a \((C, \delta)-\)compatible automaton \( H \), where \( C \) is a specific calculus method chosen for Botchkarev’s algorithm.
Theorem 7.2.5 Let $H$ be a $(\mathcal{C}, \delta)$-compatible hybrid automaton and $R_I$ be an initial region for $H$. Furthermore, let $l$ be a real value in $\mathbb{R}_{>0}$ and $R_T = \langle v_T, \phi_T \rangle$ be a set of states such that $[R_T]_p$ is compact and convex. Then Botchkarev’s algorithm, applied to $H$, $R_I$, and $R_T$, with maximum flow time $t$ and time step $\delta$, performs at most $\psi_{l,\delta}(n)$ cycles, where $n$ is the maximum number of arcs leaving a location and:

$$
\psi_{l,\delta}(n) \overset{\text{def}}{=} \begin{cases} 
\left( \frac{t}{\delta} \right) + 1 + 1 & \text{if } n = 1, \\
\frac{n + 1}{n-1} + 1 & \text{if } n \neq 1.
\end{cases}
$$

Proof. By the definition of $(\mathcal{C}, \delta)$-compatibility, let $H$ be an $\epsilon_H$-expansive automaton and $\gamma$ be the approximation index of the calculus method used by Botchkarev’s algorithm. Thus, by definition of $(\mathcal{C}, \delta)$-compatibility, for each edge $e = \langle v, v' \rangle$:

$$
\epsilon_H - \gamma > \left( e^{SN_{A'}^w} - I \right) D(e) + e^{SN_{A'}^w} N_{U^w} \delta.
$$

Let $\langle R, t_R \rangle$ be a pair selected from $T$ during the computation, with $R = \langle v', \phi \rangle$. If $\langle R, t_R \rangle$ was inserted in $T$ due to a reset, then $T_{\delta,k}(R) + B_{e,v^w}(0)$, where $\epsilon_{v^w, \delta}$ is computed as reported in Lemma 7.1.1, does not intersect the activation of any edge $e_i = \langle v^i, v'' \rangle$ and thus $j_{\min(e_i)} > 0$, for each $i \in [1, l]$.

Moreover, when $t_R > \bar{t}$ and then $m = \lceil \frac{t-t_R}{\delta} \rceil = 0$, at step 2(c)ii of the algorithm, it holds:

$$
\bigcup_{j = j_{\min(e_i)}}^m (S_{w,j}^+ \cap A(a_i)) = \emptyset, \text{ for each arc } e_i;
$$

hence $V_{e_i}^+ = W_{e_i}^+ = \emptyset$, and so no pair is inserted in $T$.

Hence, during a generic cycle of the algorithm different from the first one, if the pair $\langle R, t_R \rangle$ is selected from $T$ and the algorithm does not terminate at this cycle, there are two possibilities:

- if $t_R > \bar{t}$, then no pair is inserted into the table $T$, as shown previously;
- if $t_R \leq \bar{t}$, then at most $n' \leq n$ pairs, $\langle \cdot, \tau' \rangle$, are inserted into the table $T$.

Furthermore, as we have shown, $j_{\min(e_i)} > 0$, for each $i \in \{1, \ldots, l\}$, $\tau' \geq \tau + \delta$.

From the facts that the algorithm can insert in $T$ at most $n$ pairs in the first cycle and that $\tau' \geq \tau + \delta$, it follows that the maximum number of cycles is $n \ast \phi_{l,\delta}(0, n) + 1$, where:

$$
\phi_{l,\delta}(\tau, n) \overset{\text{def}}{=} \begin{cases} 
n \ast \phi_{l,\delta}(\tau + \delta, n) + 1 & \text{if } \tau + \delta \leq \bar{t} \\
1 & \text{otherwise}
\end{cases}
$$

We distinguish three cases:

$n = 0$ There is no arc and so the algorithm ends after a cycle.
n = 1 Then:
\[ \phi_{\ell,\delta}(\tau, 1) = \begin{cases} \phi_{\ell,\delta}(\tau + \delta, 1) + 1 & \text{if } \tau + \delta \leq \bar{t} \\ 1 & \text{otherwise} \end{cases} \]

Thus, \( \phi_{\ell,\delta}(0, 1) = \left\lfloor \frac{\ell}{\delta} \right\rfloor + 1 \).

n > 1 By solving the above recurrence, the maximum number of cycles is:
\[ \phi_{\ell,\delta}(0, n) = \left\lfloor \frac{n}{\delta} \right\rfloor + 1 \]
\[ = \frac{n\left\lfloor \frac{\ell}{\delta} \right\rfloor + 1 - 1}{n - 1} \]

Hence,
\[ \phi_{\ell,\delta}(0, n) = \begin{cases} \left\lfloor \frac{n}{\delta} \right\rfloor + 1 & \text{if } n = 1 \\ \frac{n\left\lfloor \frac{\ell}{\delta} \right\rfloor + 1 - 1}{n - 1} & \text{if } n \neq 1 \end{cases} \]

Finally, the maximum number of cycles performed by the algorithm, \( \psi_{\ell,\delta}(n) = n * \phi_{\ell,\delta}(0, n) + 1 \), is given by:
\[ \psi_{\ell,\delta}(n) = \begin{cases} n\left( \left\lfloor \frac{n}{\delta} \right\rfloor + 1 \right) + 1 & \text{if } n = 1, \\ n\left( \frac{n\left\lfloor \frac{\ell}{\delta} \right\rfloor + 1 - 1}{n - 1} \right) + 1 & \text{if } n \neq 1. \end{cases} \]

From this theorem, the following termination result follows.

**Corollary 7.2.6** Let \( H \) be a \( (\mathcal{C}, \delta) \)-compatible hybrid automaton and \( R_I \) be an initial region for \( H \). Furthermore, let \( \bar{t} \) be a real value in \( \mathbb{R}_{>0} \) and \( R_T = (v_T, \varphi_T) \) be a set of states such that \( [R_T]_\omega \) is compact and convex. If \( R_T \) is reachable by \( H \) in time \( t \leq \bar{t} \), then Botchkarev’s algorithm, applied to \( H, R_I \) and \( R_T \), with maximum flow time \( \bar{t} \) and time step \( \delta \), terminates and returns REACHABLE.

**Proof.** It follows directly from Theorem 7.2.5.

**7.3 An Improved Approximation Algorithm**

The approximation of the flow tube made by Botchkarev’s algorithm tends to be too conservative because step 2(c)ii may include in the reachable set many unreachable states, which we call trash states. For instance, consider the automaton in Figure 7.2 and suppose we want to evaluate the states that are flow-reachable from \( R_I = (v, \varphi_I) \), where \( \varphi_I[Z] \equiv Z_1 = 0 \land Z_2 = -1 \). The result of applying Botchkarev’s algorithm at the end of step 2a is shown in Figure 7.3(a). The set indicated as \( V_c^+ \) in the
7.3. An Improved Approximation Algorithm

\[
\begin{align*}
\dot{Z}_1 &= -Z_2, \\
\dot{Z}_2 &= Z_1
\end{align*}
\]

\[
\text{Act}(e)[Z] \equiv \text{tt}; \\
\text{Reset}(e)[Z, Z'] \equiv \varphi_j[Z]
\]

Figure 7.2: A critical example for Botchkarev’s approximation algorithm: many unreachable states are classified as reachable.

The algorithm is the ellipsoid shown in Figure 7.3(b). The grey area in Figure 7.3(c) is the set of all unreachable states, which are classified as reachable because of the over-approximation of the union made at step 2(c)ii.

To decrease the number of trash states, we propose a new algorithm that avoids performing the union operation. This algorithm is more expensive computationally, but it obtains a tighter flow-tube approximation and has a better theoretical approximation index than Botchkarev’s one. The new algorithm is based on the fact that a state \( \ell \) is reachable from the set \( \bigcup_{j=\min(e_i)}^{m} (S^+_{v,j} \cap A(e_i)) \) if and only if \( \ell \) is reachable from the set \( S^+_{v,j} \cap A(e_i) \), for some \( j \in [\min(e_i), m] \). Thus, we can evaluate the reach set of \( \bigcup_{j=\min(e_i)}^{m} (S^+_{v,j} \cap A(e_i)) \) as the union of the reach sets of each \( S^+_{v,j} \cap A(e_i) \).

This idea could be implemented using a recursive function called \( \text{verify} \) whose parameters are \( R, R_T, t_R, \bar{t}, \) and \( \delta \), where \( R \) and \( R_T \) are state region, while \( t_R, \bar{t} \) and \( \delta \) are real numbers. This function returns \( \text{TRUE} \), if any state in \( R_T \) is classified as reachable from \( R \) in time \( (\bar{t} - t_R) \).

The main function of the algorithm does the following.

1. if \( \text{verify}(R_I, \bar{t}, 0, R_T, \delta) = \text{TRUE} \), where \( R_I \) is an initial region for \( H \), then return \( \text{REACHABLE} \);
2. return \( \text{UNREACHABLE} \).
The function \( \text{verify}(R, \bar{t}, t_R, R_T, \delta) \) performs the following steps.

1. let \( R \) be the region \( R = (v, S) \). For each \( j \in \{0, \ldots, \left\lceil \frac{\bar{t} - t_R}{\delta} \right\rceil \} \):
   
   
   (a) evaluate \( S_{v,j}^+ = ([T_{\delta,j} (R)]_{\bar{t}} + B_{t,v,j} (\emptyset)) \cap \mathcal{I}(v) \);
   
   (b) let \( R_T \) be the region \( (v_T, \varphi_T) \). If \( v = v_T \) and \( S_{v,j}^+ \cap [R_T]_{\bar{t}} \neq \emptyset \) for some \( j \)
   then return \( \text{TRUE} \);
   
   (c) for each edge \( e = (v, v') \):
      
      i. if \( S_{v,j}^+ \cap A(e) = \emptyset \), then set \( V_{v,j}^+ = \emptyset \), else evaluate the tightest over-approximation \( V_{v,j}^+ \) of \( S_{v,j}^+ \cap A(e) \) in used calculus method.
      
      ii. if \( V_{v,j}^+ \neq \emptyset \):
          
          A. evaluate the tightest over-approximation \( R_{v,j}^+ \) of \( [(v, V_{v,j}^+)]_e^- \);
          
          B. if \( \text{verify}(R_{v,j}^+, t, t_R + j\delta, R_T, \delta) \) returns \( \text{TRUE} \), then return \( \text{TRUE} \);

2. return \( \text{FALSE} \).

Although termination is not guaranteed for this new algorithm either, we can prove a complexity bound analogous to Theorem 7.2.5.

**Theorem 7.3.1** Let \( H \) be a \((C, \delta)\)-compatible hybrid automaton and \( R_I \) be an initial region for \( H \). Furthermore, let \( \bar{t} \) be a real value in \( \mathbb{R}_{>0} \) and \( R_T = (v_T, \varphi_T) \) be a set of states such that \([R_T]_{\delta} \) is compact and convex. The proposed algorithm, applied to \( H, R_I, \) and \( R_T \), with maximum flow time \( t \) and time step \( \delta \), calls the function verify at most \( \tilde{\psi}_{t,\delta}(n) \) times, where \( n \) is the highest number of arcs leaving a location and:

\[
\tilde{\psi}_{t,\delta}(n) \overset{\text{def}}{=} (n + 1) \left\lceil \frac{t}{\delta} \right\rceil.
\]

**Proof.** The number \( \tilde{\psi}_{t,\delta}(R, t_R) \) of recursive calls to compute \( \text{verify}(R, \bar{t}, t_R, R_T, \delta) \) is the sum of the calls to evaluate \( \text{verify}(R_{v,j}^+, t, t_R + j\delta, R_T, \delta) \), for each edge \( e = (v, v') \), where \( R = (v, \varphi_R) \), and index \( j \), plus the initial call itself \( \text{verify}(R, \bar{t}, t_R, R_T, \delta) \):

\[
\tilde{\psi}_{t,\delta}(R, t_R) = \sum_{e = (v, v') \in E} \sum_{j = j_{\min(e)}}^{\left\lceil \frac{\bar{t} - t_R}{\delta} \right\rceil} \tilde{\psi}_{t,\delta}(R_{v,j}^+, t_R + j\delta) + 1
\]

\[
\leq n \left( \max_{e = (v, v') \in E} \sum_{j = j_{\min(e)}}^{\left\lceil \frac{\bar{t} - t_R}{\delta} \right\rceil} \tilde{\psi}_{t,\delta}(R_{v,j}^+, t_R + j\delta) \right) + 1,
\]

where \( n \) is the maximum numbers of transitions that leave a given location.

By the compatibility of the automaton, we deduce that \( j_{\min(e)} \geq 0 \) in the first call of \( \text{verify}(R_I, \bar{t}, 0, R_T, \delta) \), and that \( j_{\min(e)} > 0 \) in the following recursive calls. Thus, if
7.3. An Improved Approximation Algorithm

\( \tilde{\psi}_{t, \delta}(n) \) is the maximum number of recursive calls to compute \( \text{verify}(R_I, t, 0, R_T, \delta) \), the following recursive definition holds:

\[
\tilde{\psi}_{t, \delta}(n) = n \left( \tilde{\phi}_{t, \delta}(0, n) + \sum_{j=1}^{\left\lceil \frac{t}{\delta} \right\rceil} \tilde{\phi}_{t, \delta}(j \ast \delta, n) \right) + 1
\]

where:

\[
\tilde{\phi}_{t, \delta}(t, n) \overset{\text{def}}{=} n \left( \sum_{j=1}^{\left\lceil \frac{t}{\delta} \right\rceil} \tilde{\phi}_{t, \delta}(t + j \ast \delta, n) \right) + 1
\]

Let us introduce the auxiliary function \( \hat{v}(i, n) \):

\[
\hat{v}(i, n) \overset{\text{def}}{=} n \left( \sum_{j=1}^{i} \hat{v}(i - j, n) \right) + 1
\]

We show next by induction on \( i \) that, for \( i \geq 0 \), \( \hat{v}(i, n) = (n + 1)^i \).

\( i = 0 \) By definition of \( \hat{v}(i, n) \), \( \hat{v}(0, n) = 1 = (n + 1)^0 \).

\( i > 0 \) By inductive hypothesis we know that \( \forall i' < i \quad \hat{v}(i', n) = (n + 1)^{i'} \). Thus, by definition of \( \hat{v}(i, n) \):

\[
\hat{v}(i, n) = n \left( \sum_{j=1}^{i} \hat{v}(i - j, n) \right) + 1
\]

\[
= n \left( \sum_{j=0}^{i-1} \hat{v}(j', n) \right) + 1
\]

\[
= n \left( \frac{(n + 1)^{i-1} - 1}{(n + 1) - 1} \right) + 1
\]

\[
= n \left( \frac{(n + 1)^{i-1} - 1}{n} \right) + 1
\]

\[
= (n + 1)^i
\]

Moreover, if we set \( k \overset{\text{def}}{=} \left\lceil \frac{t}{\delta} \right\rceil \), it is easy to show that \( \tilde{\phi}_{t, \delta}(\tau, n) = \hat{v}(k, n) \). Thus:

\[
\tilde{\phi}_{t, \delta}(0, n) = (n + 1)^{\left\lceil \frac{t}{\delta} \right\rceil}
\]

Furthermore, it follows that:

\[
\tilde{\psi}_{t, \delta}(n) = n \left( \sum_{j=0}^{\left\lceil \frac{t}{\delta} \right\rceil} \tilde{\phi}_{t, \delta}(j \ast \delta, n) \right) + 1
\]
\[ = n \tilde{\phi}_{t, \delta}(0, n) + n \left( \sum_{j=1}^{\lceil \frac{n}{\delta} \rceil} \tilde{\phi}_{t, \delta}(j \ast \delta, n) \right) + 1 \]

Hence, by definition of \( \tilde{\psi} \):
\[
\tilde{\psi}_{t, \delta}(n) = n \tilde{\phi}_{t, \delta}(0, n) + \tilde{\phi}_{t, \delta}(0, n)
\]
\[
= (n + 1) \tilde{\phi}_{t, \delta}(0, n)
\]

Furthermore, since \( \tilde{\phi}_{t, \delta}(0, n) = (n + 1) \lceil \frac{n}{\delta} \rceil \) as computed above:
\[
\tilde{\psi}_{t, \delta}(n) = (n + 1) (n + 1) \lceil \frac{n}{\delta} \rceil
\]
\[
= (n + 1) \lceil \frac{n}{\delta} \rceil + 1
\]

To gauge the accuracy of the new algorithm, we introduce a new kind of index called algorithmic approximation index. This new index gives an error estimate of the flow-tube approximation to compare the quality of different reachability algorithms.

**Definition 7.3.2 (Algorithmic Approximation Index)** Let \( H \) be a hybrid automaton. Moreover, let \( C \) be a calculus method and \( \alpha_C \) be an approximating algorithm which use \( C \) to decide the reachability of a region \( R_T \) from a region \( R_I \). Furthermore, for each pair of subsequent edges, \( e = \langle v_1, v_2 \rangle \) and \( e' = \langle v_2, v_3 \rangle \), let \( F^+_{e, e'}^{\alpha_C} \) be the over-approximation made by \( \alpha_C \) of the set
\[
F_{e, e'} = [(R_e)^-]_e \cap A(e'),
\]
where \( R_e \) is the region \( \langle v_2, \text{Reset}(e) \rangle \). The algorithmic approximation index of \( \alpha \) on \( H \) is the real number
\[
\Upsilon_H^\alpha = \max_{(e, e') \in S_h((V, E))} h^+ \left( F^+_{e, e'}^{\alpha_C}, F_{e, e'} \right).
\]

Note that the smaller is the approximation index of an algorithm, the better is the precision in flow-tube evaluation. As a matter of fact, let us consider two algorithms, \( \alpha \) and \( \beta \) that check reachability of the same set of states with the same maximum flow time. Suppose that they have algorithmic approximation indices at time \( t \) equal to \( \Upsilon_H^\alpha \) and \( \Upsilon_H^\beta \), where \( \Upsilon_H^\alpha \leq \Upsilon_H^\beta \), respectively. If \( \alpha \) verifies reachability then \( \beta \) does too. On the other hand, if \( \beta \) verify reachability, \( \alpha \) may return UNREACHABLE. Therefore, the set of unreachable states that algorithm \( \alpha \) marks as REACHABLE is a subset of the states that \( \beta \) marks as REACHABLE. For this reason, it is better to use an algorithm with a low algorithmic approximation index.

The following theorem states that the approximation index of the new algorithm is smaller than the one of Botchkarev’s algorithm; thus, it produces a tighter approximation of the reached set.
Theorem 7.3.3 Let $H$ be a linear hybrid automaton. If $\Upsilon_H^\alpha$ and $\Upsilon_H^\beta$ are the algorithmic approximation indices on $H$, respectively, of the algorithm presented at page 101 and of Botchkarev’s algorithm, then $\Upsilon_H^\alpha \leq \Upsilon_H^\beta$.

Proof. Consider the executions of both $\text{verify}(R, \bar{t}, t_R, R_T, \delta)$ and of Botchkarev’s algorithm, assuming that the latter selected the pair $(R, t_R)$ from the table $T$. Both algorithms evaluate the same sets of states $S_{v,j}^+$, however, for any edge $e = (v, v')$, Botchkarev’s algorithm approximates the union of all the $S_{v,j}^+$ by means of a set $V_e^+$, whereas the proposed algorithm follows the evolution of each single set $S_{v,j}^+$ instead of their union.

In particular, for any edge $e = (v, v')$, the proposed algorithm uses directly the intersection between the flow-tube approximation and the activation of $e$:

$$V_e = \left[\frac{t - t_R}{\delta_m}\right] \bigcup_{j = j_{\min(e)}} (S_{v,j}^+ \cap \mathcal{A}(e)),$$

whereas the original algorithm of Botchkarev over-approximates it with a set $V_e^+ \supseteq V_e$.

It follows that the algorithmic approximation indices of the proposed algorithm and of Botchkarev’s one are

$$\Upsilon_H^\alpha = \max_{(e, e') \in Sb((V, E))} h_+\left(V_e, [F_{e, e'}]_{\bar{t}}\right)$$

and

$$\Upsilon_H^\beta = \max_{(e, e') \in Sb((V, E))} h_+\left(V_e^+, [F_{e, e'}]_{\bar{t}}\right)$$

respectively.

Furthermore, for any three sets $X_1, X_2, Y \in \mathbb{R}^n$ such that $X_1 \supseteq X_2$, the Hausdorff semi-distance satisfies the property

$$h_+ (X_1, Y) \geq h_+ (X_2, Y)$$

In particular, the above equation holds when $X_1 = V_e^+$, $X_2 = V_e$ and $Y = [F_{e, e'}]_{\bar{t}}$ for any pair of subsequent edges $e = (v_1, v_2)$ and $e' = (v_2, v_3)$. Thus, $\Upsilon_H^\alpha \leq \Upsilon_H^\beta.$

Thus, even though the calculus method used by both algorithms cannot express exactly the union of sets $S_{v,j}^+ \cap \mathcal{A}(e)$, the algorithmic approximation index of the proposed algorithm is at least as small as the one of Botchkarev’s algorithm.

7.4 Test Case Results

To test the quality of the proposed algorithm, we verified the behaviour of an engine controller. The objective was to prove that for the given engine model the controller under test maintains a crankshaft speed between 750 and 850 rotations per minutes, for a least 0.5 seconds. For that purpose, we modelled the engine and its controller.
using hybrid automata. Furthermore, we defined forbidden regions of the controlled system and we checked whether they could be reached, by using both Botchkarev’s algorithm and the one proposed here.

![Figure 7.4: Engine model](image)

We used the engine model presented in [16, 17, 18]. It is composed of three main blocks: the intake manifold, the cylinders and the powertrain. Their coupled dynamics can be described as follows.

The intake manifold’s pressure, \( p \), can be controlled by the throttle valve angle, \( \alpha \), which determines the air-fuel mixture, \( m \), loaded by the cylinders. In particular:

\[
\dot{p}(t) = a_p p(t) + b_p \alpha(t)
\]

where \( a_p \) and \( b_p \) are parameters which depend on geometric features of the intake

![Figure 7.5: Engine controller model](image)
manifold. The torque, $T$, generated by cylinders depends on the air-fuel mixture loaded by the intake manifold and on the spark ignition time. If the crankshaft rotation speed, $n$, lies between $n - \Delta_n$ and $n + \Delta_n$, with $\Delta_n$ small enough, we may assume that the mass of air-fuel loaded depends only on the intake manifold’s pressure. Thus:

$$m = k_p,$$

where $k$ is an appropriate parameter. Furthermore, if the air-fuel mixture loaded in each cylinder is always maintained in an optimal stoichiometric relation, the torque generated can be described by the following equation:

$$T(t) = Gm\eta(\phi),$$

where $\phi$ is the spark advance, $G$ is an appropriate parameter and $\eta(\phi) = \frac{1}{175}(2\phi + 135)$ is the ignition efficiency function.

### Table 7.1: Flow equations of the model

(a) Flow equations in $S$ and $S^+$ (the clutch is on).

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\dot{n}(t) = an(t) + bT(t)$</td>
<td>\dot{n}(t) = a_n(t) + b_T(t)</td>
</tr>
<tr>
<td>$\dot{\theta}(t) = 6n(t)$</td>
<td>\dot{\theta}(t) = 6n(t)</td>
</tr>
<tr>
<td>$\dot{T}(t) = 0$</td>
<td>\dot{T}(t) = 0</td>
</tr>
<tr>
<td>$\dot{\phi}(t) = 0$</td>
<td>\dot{\phi}(t) = 0</td>
</tr>
<tr>
<td>$\dot{m}(t) = 0$</td>
<td>\dot{m}(t) = 0</td>
</tr>
<tr>
<td>$\dot{p}(t) = a_p\rho(t) + b_p\alpha(t)$</td>
<td>\dot{p}(t) = a_p\rho(t) + b_p\alpha(t)</td>
</tr>
</tbody>
</table>

(b) Flow equations in $S_L$ and $S_L^+$ (the clutch is released).

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\dot{n}(t) = a_Ln(t) + b_LT(t)$</td>
<td>\dot{n}(t) = a_Ln(t) + b_LT(t)</td>
</tr>
<tr>
<td>$\dot{\theta}(t) = 6n(t)$</td>
<td>\dot{\theta}(t) = 6n(t)</td>
</tr>
<tr>
<td>$\dot{T}(t) = 0$</td>
<td>\dot{T}(t) = 0</td>
</tr>
<tr>
<td>$\dot{\phi}(t) = 0$</td>
<td>\dot{\phi}(t) = 0</td>
</tr>
<tr>
<td>$\dot{m}(t) = 0$</td>
<td>\dot{m}(t) = 0</td>
</tr>
<tr>
<td>$\dot{p}(t) = a_p\rho(t) + b_p\alpha(t)$</td>
<td>\dot{p}(t) = a_p\rho(t) + b_p\alpha(t)</td>
</tr>
</tbody>
</table>

The powertrain dynamics depends both on the crankshaft speed, $n$, and the angle, $\theta$, between the current crankshaft’s position and the position of the previous dead center. This relation can be modelled by:

$$\begin{align*}
\dot{n}(t) &= an(t) + bT(t) \\
\dot{\theta}(t) &= k_n n(t),
\end{align*}$$

where $a$ and $b$ are parameters that depend on the the position of the clutch. For this reason it is necessary to distinguish between two different systems: the former models the powertrain when the clutch is on, while the latter describes the powertrain when the clutch is released.

The overall engine model is depicted in Figure 7.4 and the corresponding flow equations and initial conditions are reported in Table 7.1. This engine model was used to verify the correct behaviour of the controller model shown in Figure 7.5.

The proposed algorithm was implemented as part of a revised version of Verishift, a software package for reachability analysis of linear hybrid automata written by Botchkarev and Tripakis [27, 28], based on Botchkarev’s original algorithm. The modified version of Verishift and the models presented in Section 7.4 are available
Table 7.2: Initial conditions of the model

\[ n(0) \in [799, 801] \quad n_0 = 800 \text{ rotation per minute} \]

\[ \theta(0) = 0^\circ \quad T(0) = -\frac{a}{b} n(0) = 12.81 \text{ Nm} \]

\[ \phi(0) = 20^\circ \quad m(0) = \frac{T(0)}{C} = -\frac{a}{b} \frac{n(0)}{C} = 9.286 \times 10^{-5} \text{ Kg} \]

\[ p(0) = \frac{m(0)}{\kappa} = 7300 \text{ Pa} \quad \alpha(0) = \frac{\alpha}{C} p(0) = 7.02^\circ \]

at [http://fsv.dimi.uniud.it/papers/improving_EC2004](http://fsv.dimi.uniud.it/papers/improving_EC2004). Although Botchkarev’s algorithm reported as reachable some “bad” regions of the controlled system, the new algorithm proved that such regions actually are not reachable. A partial representation of the model’s flow tube is given in Figure 7.6.

Figure 7.6: Flow tube of the case study. Only one branch of the recursive computation tree is reported.
ARIADNE - A library for hybrid automata

“Five is a sufficiently close approximation to infinity.”

R. Firth

Various tools, implementing ideas discussed in the previous chapters, have been proposed in the last decade. In particular, many tools, based on approximation techniques [74, 54, 12, 88, 28, 128], have been developed to allow the study of hybrid automata for which reachability is undecidable. Among the most important are HyTech [80], d/dt [11], Checkmate [127], UPPAAL [26], and KRONOS [55]. These tools also include many interesting features such as model checking capabilities or graphical modelling interfaces. However, they have two main disadvantages. First, these software packages have restrictive licenses, and some are even closed source. Without access to the source code, users can neither customise or optimise them for a specific class of instances of the reachability problem, nor check that the algorithms are correctly implemented. Second, these software do not support arbitrary-precision evaluation of the reachable sets. As a matter of fact, Collins and Lygeros proved in [52] that, despite the use of arbitrary-precision data structures, arbitrary-precision approximations of reachable sets are not computable in general.

To overcome these limitations, we propose a framework for hybrid system verification, with three main goals. The first goal is to specify a development environment in which to construct space representation techniques and algorithms for reachability analysis. The second goal is to build a tool which integrates and implements existing algorithms and representation techniques, to let users choose the best methods for their needs. The last goal is to support arbitrary-precision approximate representations of the hybrid automaton evolutions.

This meta-tool, called ARIADNE, offers an unifying high-level scheme based on general modules designed to be easily tuned for various requirements. The package will be released as an open source distribution, so that different research groups may contribute new data structures, algorithms and heuristics. ARIADNE implements
All calculus methods, parametric on the real domain, based on simplex, parallelo-
topes, zonotopes, and polyhedra. Moreover, it lets the user to define multivalued
functions from the set of object denotable by a calculus method to itself and it fur-
nishes evaluation methods for such functions.

8.1 Computable Analysis and Approximation

In this section we briefly present some notions from computable analysis (see [140, 51])
underlying the geometric representation and function evaluation routines adopted in
ARIADNE.

Since computers are finite state machines, they cannot represent exactly all the
objects of a continuous space, but have to resort to approximations instead. Tradi-
tionally, computers approximate real numbers using fixed-precision floating point
numbers, such as the IEEE double-precision type double. While these numbers pro-
vide sufficient accuracy for most purposes, for some particularly sensitive problems,
higher precision may be needed. Therefore we have to work with arbitrary-precision
objects.

A further issue is that errors which occur in fixed-precision floating-point compu-
tations are hard to control exactly. Interval arithmetic provides error bounds, but
these are typically very poor.

Computable analysis provides a theoretical basis for arbitrary-precision represen-
tations of points, sets and functions. It also provides a semantics for a valid compu-
tation on objects via their representations to be implemented by tool developers.

As proved by Collins and Lygeros in [52], in general, arbitrary-precision approx-
imation of reachable sets is not computable. In particular, if invariants, activations
and the reachable set are represented by closed sets, then arbitrary-precision over-
approximations of the reachable set itself are not computable in general. Symmet-
rically, if invariants, activations and the reachable set are represented by open sets,
then arbitrary-precision under-approximations of the reachable set itself are not com-
putable in general. A simple intuition of these result is presented in Example 8.1.1.

For these reasons, we exploit computable analysis to implement a tool which will
decide whenever a certain approximation can be achieved or not and which, in the
first case, will evaluate the reachable sets with that approximation.

Example 8.1.1 Let $H = (Z, Z', V, E, Inv, Dyn, Act, Reset)$ be a hybrid automaton
of dimension 1 having one location $v$ and one edge $e = (v, v)$ only. Moreover, let
$Inv, Dyn,$ and $Act$ be such that $Inv(v)[Z] \overset{\text{def}}{=} \text{tt}, Dyn(v)[Z, Z', T] \overset{\text{def}}{=} Z = Z'$, and
$Act(e)[Z] \overset{\text{def}}{=} Z \geq 2$, respectively. Finally, let us assume that we would like to compute
the reachable set of $H$ from $Sat(\iota)$, where $\iota \overset{\text{def}}{=} Z < 2$. It is easy to verify that the
reachable set is $Sat(\iota)$ itself. However, every closed set including Sat(\iota) includes
Sat(\bar{\iota}), where \bar{\iota} \overset{\text{def}}{=} Z \leq 2, also. Hence, if we try to over-approximate reachable set
using close sets, the computed over-approximation intersects the $e$'s activation and,
thus, its precision is bounded from below by $\max_{Z \in A(e), Z' \in R(e)}(\|Z - Z'\|)$. \(\blacksquare\)
There are two equivalent descriptions of objects provided by computable analysis: by properties and by approximations.

An open predicate on a space $X$ is a predicate $\varphi$ such that $\varphi$ is true on an open subset of $X$. A property of an object $x \in X$ is an open predicate satisfied by $x$. For example, an open predicate on $\mathbb{R}$ is given by $\varphi(x) := x < 2 \lor 3 < x < 5$, which is a property of $\pi$. An example of an open predicate on the space of continuous functions on $\mathbb{R}$ is $\varphi(f) := [0, 1] \subset f^{-1}(1/3, 2/3)$, which is a property of the function $f(x) = (x + 2)/5$.

An object can be specified by listing its properties. For example, listing all open intervals with rational endpoints containing $\pi$ is enough to specify $\pi$ completely. However, $\pi$ can also be specified by a sequence of rational open intervals $(a_n, b_n) \ni \pi$ such that $b_n - a_n \to 0$ as $n \to \infty$.

An approximation to an object $x$ is a denotable object $\xi$ which is “close” to $x$, as given by some error bound. By denotable, we mean that $\xi$ is an element of a countable dense set which can be specified by a finite amount of data. The error may be given to some accuracy relative to some metric, $d(\xi, x) < \epsilon$, but may also be given by some (partial) order, such as an under approximation $\xi < x$ or an over approximation $\xi > x$. For spaces which are both metric and ordered, we can also consider lower and upper approximations. An object can be specified by giving a convergent sequence of approximations together with error bounds.

In ARIADNE, we deal mainly with approximations, and also we may be able to verify a property of the reachable set without computing it entirely.

### 8.1.1 Representation of Points and Sets

The rational numbers $\mathbb{Q}$ and dyadic\(^1\) numbers $\mathbb{Q}_2$ both form dense subsets of the real line. We can hence use both these types as denotable real numbers. Since the real numbers are a totally-ordered metric space, we can consider approximations to a given accuracy, under-approximations and over-approximations.

Similarly, $\mathbb{Q}^n$ and $\mathbb{Q}_2^n$ can be used to approximate points in $\mathbb{R}^n$, with error given by one of the standard norms, such as the supremum norm $\|x - \xi\|_\infty = \sup\{|x_i - \xi_i|\}$ or the Euclidean norm $\|x - \xi\|_2 = (\sum |x_i - \xi_i|^2)^{1/2}$.

Any compact subset of the Euclidean space $\mathbb{R}^n$ can be approximated by a finite union of simple sets, such as intervals, simplices, cuboids, parallelepipeds, zonotopes, polyhedra, spheres and ellipsoids, each defined using dyadic or rational coefficients. Since these simple sets form a base for the topology, they are called basic sets. In ARIADNE, we use compact basic sets, denoted $\mathcal{T}$, $\mathcal{J}$, etc.. The sets that can be represented exactly as a finite union of basic compact sets (of a given type) are called denotable sets.

We can build up predicates and operations on denotable sets from the corresponding predicates and operations on basic sets. Each basic set type should support the standard predicates:

- **containment**: $x \in \mathcal{T}$,

---

\(^1\) A dyadic number has the form $m/2^n$ for $m, n \in \mathbb{Z}$.
Figure 8.1: Examples of basic sets.

- intersection: \( I \cap J \neq \emptyset \), and
- subset: \( I \subset J \).

Additionally, each basic set type should support the following “robust” predicates:

- interior containment: \( x \in I \),
- disjointness: \( I \cap J \neq \emptyset \),
- intersection of interiors: \( I \cap J \neq \emptyset \), and
- inner subset: \( I \subset J \).

These predicates are important since if true, they remain true under sufficiently small perturbations. We also wish to consider the operations of

- regular intersection: \((I, J) \mapsto cl(I \cap J)\),
- intersection: \((I, J) \mapsto I \cap J\), and
- convex hull: \((I, J) \mapsto \text{conv}(I, J)\),

where defined.

For denotable sets, we can support all the above predicates, and also

- union: \((A, B) \mapsto A \cup B\),
- upper set-difference: \((A, B) \mapsto A \setminus \text{int}(B)\), and
- lower set-difference: \((A, B) \mapsto \text{cl}(A \setminus B)\).

There are many types of approximations to compact sets which we consider. These can be defined in terms of set inclusion, and the \(\epsilon\)-neighbourhood of a set \(A\), which is the set

\[ N_\epsilon(A) = \{ x \in X \mid \exists x' \in A, \ d(x, x') < \epsilon \} \]

consisting of points which lie within \(\epsilon\) of some point of \(A\). We have the following approximations \(A'\) to a compact set \(A\):
8.1. Computable Analysis and Approximation

Figure 8.2: Examples of denotable sets.

- **$\epsilon$-lower approximation**: $A' \subset N_\epsilon(A)$,
- **over-approximation**: $A' \supset A$,
- **$\epsilon$-accurate approximation**: $A \subset A' \subset N_\epsilon(A)$.

Note that the $\epsilon$-accurate approximation is a combination of $\epsilon$-lower approximation and over-approximation. We use $\epsilon$-lower approximations since for general closed sets, under-approximations (i.e., $A' \subset A$) need not exist; as an example, the singleton $\{\pi\}$ has no under-approximation by a rational interval.

Figure 8.3: An example of approximating set.

8.1.2 Representation of Functions and Operations

By the theory presented in [140], we can specify a continuous function $f : X \rightarrow Y$ by giving a routine which computes, for all denotable $x \in X$ and $\epsilon > 0$, a denotable element $y \in Y$ and $\delta > 0$ such that $d(f(x'), y) < \epsilon$ whenever $d(x, x') < \delta$. Note that this specification mimics the standard $\delta - \epsilon$ definition of continuity. For many
functions, including polynomials and elementary functions, writing such a routine is possible. From such a routine, it is possible to compute the image of a compact set, or integrate a vector field.

For the dyadic and rational numbers, we can do better, since addition, subtraction and multiplication are exact. Division is also exact for rationals, but not for dyadic numbers, unless we divide by a power of two. Note that this is different from IEEE double-precision floating point numbers, for which even addition can only be performed approximately.

8.2 Ariadne’s Implementation

ARIADNE is a library for computation with hybrid automata under development by a joint team including University of Udine, PARADES and CWI. The goal is to build an open and easily extensible package that features basic data structures and operators to support analysis and synthesis of systems described with hybrid automata.

The ARIADNE computational kernel is written using generic programming, in which mathematical concepts, such as real numbers and continuous functions, can be implemented by different concrete types presenting the same (or similar) interfaces. In this way, we can write algorithms which work with any type having the same interface, and even if a type does not implement the full interface, some algorithms may still be available for that type. The advantage of this approach is that it greatly facilitates extensions: new types can be freely added as long as they conform to the syntax and semantics of the concept; new algorithms can be plugged-in to work with the new types, and user-defined algorithms can replace the algorithms supplied with the tool.

From the syntax and semantics of hybrid automata, we notice that any technique trying to decide the reachability problem needs:

- To represent sets of states and to apply basic geometric operations over them (i.e. intersection, union, emptiness check, etc.);
- To integrate differential equations and to apply maps to sets of states;

Thanks to these observations, we may identify two main modules in the computational kernel of a generic hybrid system reachability analysis tool: the geometry module and the evaluation module.

Moreover, we need to consider two more modules: the input module and the output module. Finally, some important types and functions used by more than one module, or imported from external libraries, are best included in a base module, and some specialised implementation techniques (data caching and parallel processing) in a storage module and a distribution module.

Hence, the overall structure of ARIADNE includes six core modules:

- Base module: widely-used fundamental types and classes; interfaces to external libraries.
8.2. Ariadne’s Implementation

ARIADNE will also include two enhancement modules:

- **Storage module**: methods to provide smart caching of large data sets.
- **Distribution module**: methods to run the tool in parallel on a network of computers.

A high-level scheme of the library’s structure is presented in Figure 8.4. The Input and Output modules present a high-level user interface, while the Geometry and Evaluation modules form the computational kernel.

### 8.2.1 The Base Module

The base module consists of some generally-useful sub-modules.

**Input module**: translates from a modelling language (e.g., Modelica [66]) to internal data structures.

**Output module**: methods to represent the computational results either graphically or in a textual way; may be interfaced with other analysis and visualisation tools.

**Geometry module**: a general framework for representing and operating with points and sets of points.

**Evaluation module**: methods to apply transformations, integrate flows, evaluate reached regions and compute traces.
The Number Sub-Module

ARIADNE uses the `double`, `dyadic` and `rational` types for approximating real numbers. The `double` type is a fixed-precision type, and is best used when raw speed is important, whereas `dyadic` and `rational` are arbitrary-precision types which may be used when accuracy and error control are important. These types are currently implemented using the GNU multiple-precision library, GMP [69].

The Function Sub-Module

In this module, basic functions such as arithmetic, special functions and polynomials are defined on the real numbers. For a unary function $f$, the basic syntax is

$$ y = f(x,e,d), $$

where $e$ is the desired error in the result $y$ and $d$ is an output parameter giving the maximum error in the argument for the result to be valid. These functions are currently implemented using the MPFR library [75].

The linear algebra sub-module

In this module, basic linear algebra operations, such as solution of linear equations, matrix factorisation and linear programming are defined. These functions include interfaces to the Fortran BLAS and LAPACK libraries.

8.2.2 The Geometry Module

The geometry module specifies and contains implementations of the concepts of `Point` and `Set`, parametrised by the real number type `Real`. The `Set` concept can be refined to `BasicSet` and `DenotableSet`. Both `Point` and `DenotableSet` also have appropriate approximation types.

The point concept has the canonical implementation `Point`, which represents a point in the Euclidean space. The only approximation allowed is a metric approximation and it is described by `Approximation(e)`.

Basic set classes supported by ARIADNE include `Interval`, `Simplex`, `Cuboid`, `Polyhedron`, `Parallelepiped`, `Spheroid`, `Ellipsoid`, and `Zonotope`. A full range of conversion and approximation operations are provided, along with the robust predicates `interior_contains`, which returns `TRUE` whenever the interior of the considered set contains the set passed as parameter, `disjoint`, which checks the disjointness of two sets, `interiors_intersect`, which returns `TRUE` only if the interiors of two sets are not disjoint, and `inner_subset`, which tests the proper inclusion of two sets. Moreover, ARIADNE furnishes the standard predicates `contains`, `intersect` and `subset`, described in Section 8.1. Further, the binary operations `regular_intersection`, `intersection` and `convex_hull` should be provided if the class is closed under the operation.

A denotable set is a finite union of basic sets of the same type. ARIADNE contains four types of denotable sets.
8.2. Ariadne’s Implementation

- **ListSet<Real,BasicSet>** is parametrised by the real number type (double, dyadic or rational) and the type of basic set, and is valid for all types of basic sets.

- **GridListSet<Real>** and **GridMaskSet <Real>**, which are sets defined on cubical grids.

- **PartitionTreeSet<Real>** for sets defined on cuboidal partition trees [126].

- **SimplicialSet<Real>** for sets defined using simplicial complexes.

In addition to the operations on basic sets, denotable sets support the operations **join** which computes the union\(^2\), **lower_difference** and **upper_difference**.

Any approximation can be described using a closed set which have type either **LowerApproximation(e)** or **OverApproximation** or **Approximation(e)**. Such types correspond to the three approximation types described in Section 8.1 where the type **Approximation(e)** corresponds to the \(\epsilon\)-accurate approximation type.

### 8.2.3 The Evaluation Module

The evaluation module includes the concepts **MAP** and **VECTORFIELD**, along with algorithms **apply** for computing the image of a map (acting on a state or a set) and **integrate** for computing the flow of a vector field. From these basic types and algorithms, we can write algorithms for computing the flows of hybrid systems.

For functions on arbitrary-precision real-number types, there is no canonical accuracy, and a global precision variable gives the default approximation type. The unary operator – is used as syntactic sugar for neg, and binary operators +, –, * and / are used for add, sub, mul and div operations with a default approximation type.

Maps include arithmetical functions such as **add**, and elementary functions such as **exp**, **log** and **sin**. The definition of these functions on real numbers are imported from the function sub-module. In the evaluation module, these operations are extended to act on points and sets.

ARIADNE also allows the creation of user-defined maps and vector fields, using a standard syntax. An implementation must be provided for the action of the map on states, from which the action on sets is automatically generated. Moreover, user-defined actions on sets may be given if these are more efficient or accurate than the built-in operations.

ARIADNE includes support for integrating vector fields, including arbitrary precision and set-based integrators. If \(f\) is a vector field, \(U\) is a domain, \(C\) is an initial state set and \(T\) is a time interval, then the operator **integrate** computes the set reached at times \(t \in T\) under the vector field \(f\) with initial conditions \(C\) remaining in the set \(U\). Different classes may be used to implement the **integrate** operator, such as Runge-Kutta integrators for states, and set-based integrators for differential inclusions (see [59, 129]) including the Euler method, and a \(C^1\)-Lohner algorithm for parallelepipeds (see [143]).

\(^2\)union is a keyword in C++
Operations may be *exact* or *approximate* for a given type. Approximations are determined by the following classes, where we use $y$ to denote the exact answer, and $\eta$ the approximate result.

- **Exact**: The answer is guaranteed to be exact.
- **Nearest**: The answer is guaranteed to be the closest possible value which can be represented by the number type.
- **Ieee**: The answer is specified by an IEEE standard.
- **Under**: The answer is guaranteed to be lower than the exact value.
- **Over**: The answer is guaranteed to be higher than the exact value.
- **Approximate**: The answer is implementation-defined, and no guarantees are given.

For fixed-precision arithmetic, we have the following:

- **Nearest**: The answer is guaranteed to be the closest possible value which can be represented by the number type.
- **Ieee**: The answer is specified by an IEEE standard.
- **Under**: The answer is guaranteed to be lower than the exact value.
- **Over**: The answer is guaranteed to be higher than the exact value.
- **Approximate**: The answer is implementation-defined, and no guarantees are given.

For arbitrary-precision arithmetic, we have the following:

- **Accuracy($e$)**: The answer is guaranteed to have an error of at most $e$.
- **Under($n$)**: The answer $\eta$ is guaranteed to satisfy $\eta \leq y$ and $\eta \not\to y$ as $n \to \infty$.
- **Over($n$)**: The answer $\eta$ is guaranteed to satisfy $\eta \geq y$ and $\eta \not\to y$ as $n \to \infty$.
- **Lower($n$, $e$)**: There exists $\zeta$ such that $\eta \leq \zeta$, $d(\zeta, y) < e$, and $\eta \to y$ as $n \to \infty$ and $e \to 0$.
- **Upper($n$, $e$)**: There exists $\zeta$ such that $\eta \geq \zeta$, $d(\zeta, y) < e$, and $\eta \to y$ as $n \to \infty$ and $e \to 0$.

The parameter $n$ is used to obtain a convergent sequence when it is not possible to give a rate of convergence. Not all operations need implement all approximation types. For certain ARIADNE types, further guarantees about the answer may be required; for example, a compact set computed to accuracy $\epsilon$ should also be an over-approximation.

Currently the geometry module is substantially completed, and work is in progress on the evaluation module. These modules form the computational kernel of the tool, and once they have been completed and tested, work will begin on the input and output modules, while optimisations of the computational kernel will continue.

### 8.3 Comparison with Existing Languages/Tools

In [115] a survey of languages and tools for hybrid systems is presented, together with an analysis of requirements that should be supported by an Interchange Format. The existing tools are classified according to the approaches used in modelling hybrid systems (Table 8.1 in [115]) and to their main features (Table 8.2 in [115]). We augment Table 8.1 and Table 8.2 from [115] with the data of ARIADNE and summaries the features available in ARIADNE.
### 8.3. Comparison with Existing Languages/Tools

<table>
<thead>
<tr>
<th>Name</th>
<th>Continuous/Discrete Specification</th>
<th>State/Dynamics Mapping</th>
<th>Continuous/Discrete Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIADNE</td>
<td>defined by language modifier</td>
<td>mode refinement into continuous</td>
<td>indirect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dynamics</td>
<td></td>
</tr>
<tr>
<td>CHARON</td>
<td>defined by language modifier</td>
<td>mode refinement into continuous</td>
<td>indirect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dynamics</td>
<td></td>
</tr>
<tr>
<td>CHECKMATE</td>
<td>separation between FSMs and</td>
<td>discrete output from FSMs to</td>
<td>event generator first-order hold</td>
</tr>
<tr>
<td></td>
<td>dynamical systems</td>
<td>dynamical systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HYSDEL</td>
<td>real and Boolean signals</td>
<td>discrete output from FSMs to</td>
<td>event generator first-order hold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dynamical systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HYVISUAL</td>
<td>signal attribute/automatic type</td>
<td>state refinement into continuous</td>
<td>toContinuous</td>
</tr>
<tr>
<td></td>
<td>detection</td>
<td>models</td>
<td>toDiscrete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>actors</td>
</tr>
<tr>
<td>MODELICA</td>
<td>defined by language modifier</td>
<td>differential equation sets depending</td>
<td>indirect when</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on events</td>
<td>statements</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCICOS</td>
<td>defined by port attribute</td>
<td>differential equation set of event</td>
<td>indirect continuous/discrete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectors</td>
<td>states</td>
</tr>
</tbody>
</table>

Table 8.1: Various approaches to modelling hybrid systems (Table 8.1 from [115], enriched with ARIADNE’s data).
Object Orientation: This feature is not implemented yet, but we intend to include it in the future.

Hierarchy: This feature is not implemented yet, but we plan to include it in the final version of the tool.

Heterogeneous Modelling: Thanks to modularity, the tool can support different computational methods (based on the integration of continuous dynamics or on algebraic decidability), as well as different geometric representations (e.g., polyhedra, simplices, ellipsoids, zonotopes, and mixed ones). At the moment, we are developing a module for the integration of linear systems, together with a polyhedral representation module and an ellipsoidal representation module.

Implicit Equations: All the invariant constraints and region specifications are represented using implicit equations.

Explicit Declaration of Locations and Transitions Manager: This feature is supported.

Explicit Declaration of Invariant Constraints: The tool supports explicit declaration of invariants, in particular, any location definition must include the corresponding invariant specification.

Explicit Non-Determinism: Non-determinism is supported at the discrete transition level. No admissible trace is missed.

We do not expect intrinsic problems when translating the models of ARIADNE into the Metropolis MetaModel Format [15] and vice versa.
Table 8.2: Main features offered by the languages/tools of Table 8.1 (Table 8.2 from [115], augmented with the data of Ariadne).
8. ARIADNE - A library for hybrid automata
Conclusions

“All men are mortal. Socrates was mortal. Therefore, all men are Socrates.”

W. Allen

In this dissertation, we considered the model checking problem over hybrid automata. In particular, we exploited some well known results on both logic and analysis and we gave an example of how a tighter interaction between these two mathematical fields can still bring some interesting results about hybrid system verification. Actually, we think that to obtain further improvements in this field, we will need the collaboration of three main ingredients: logic, analysis, and computer science. Developing more efficient algorithms to decide polynomial formulae and proving the decidability of theories such as \((\mathbb{R}, 0, 1, +, *, e, \geq)\) or \((\mathbb{R}, 0, 1, +, *, (f_{f \in \mathbb{C}}, \geq))\) are fundamental aims for the future. We should identify general analysis results which allow us to reduce continuous reachability to either small formulae decidability or low complexity methods. Finally, computer science will bridge such results and it will discriminate effective algorithms from techniques which can not be used because they are not computable.

In the first part of this thesis we introduced models, temporal logics, and model checking and we presented some interesting results over them. Moreover, we formalised the notion of hybrid automaton and we described the state-of-the-art of model checking techniques over it.

In the second part of this thesis, we related mathematical analysis and logics and we proved that the existence of a function satisfying conditions expressed by first-order formulae can be reduced to a first-order satisfiability problem over the reals. Such reduction is achieved exploiting selection results for multivalued functions and we used it to reduce bounded model checking problems for hybrid automata to the first-order decidability problem. This proves a tight connection between logics and analysis in both dynamical and hybrid context and it shows a new approach to solve the model checking problem over such kind of systems. We considered hybrid automata whose dynamics are inclusion dynamics defined by first-order formulæ. In particular, we showed that even if the automaton’s dynamics are continuous, we cannot guarantee the existence of a continuous transition satisfying the dynamics themselves. For this reason, we defined a set of conditions which relates the existence of such continuous transition and the truth value of a first-order formula. Since such result is obtained using a selection theorem proved by Michael [105], we say that a hybrid automaton satisfying such conditions is in Michael’s form. If \(H\) is a hybrid automaton in Michael’s form, then we were able to write the first-order formula \(\overline{\text{Reach}}(H, ph)[Z, Z’, T]\) which holds if and only if \(H\) can reach \(Z'\) from \(Z\) in
time $T$ through a trace whose corresponding path is $ph$. Exploiting this result, we presented two interesting classes of hybrid automata in Michael’s form. The first of these classes is the class of First-Order Constant Reset automata, FOCoRe. A FOCoRe is a first-order hybrid automaton in Michael’s form whose resets are constant maps. Thanks to the constant reset condition, we proved that we can reduce the general reachability problem over any FOCoRe $H$ to a the reachability problem over traces of length at most equal the the number of $H$’s discrete edges. It follows that the reachability problem for FOCoRe is decidable. We introduced a CTL sub-logic called $\Phi_P$ and we proved that model checking problems expressed on $\Phi_P$ are decidable over FOCoRe. Notice that, since $\Phi_P$ is not preserved by simulation and since there exist FOCoRe having infinite bisimulation quotient, our decidability results cannot be achieved exploiting standard equivalence reduction techniques. The second class of hybrid automata that we introduced is the class of Independent Dynamics Automaton, IDA. In such class, we relaxed constant reset condition maintaining, under some further assumptions, the decidability of the reachability problem. In particular, a hybrid automaton is an IDA if its variables can be partitioned into two subsets, the independent variables set and the dependent variables set. We required every variable in the first set to have both identity resets and transitive dynamics and we imposed to such variables the same dynamics in all locations. Exploiting such conditions, we were able to over-estimate the time needed to reach a set $Sat(\tau)$ from the set $Sat(\iota)$. Since we also imposed to IDA’s dynamics a minimum elapsed time between two discrete transitions, if the above set is bounded, then we can compute symbolically the maximum number of discrete transitions in a generic trace from $Sat(\iota)$ to $Sat(\tau)$. Hence, we can decide reachability problem. We showed that there exist FOCoRe which are IDA also, and we proved that there exists IDA having infinite bisimulation quotient. Finally, we showed that the problem of deciding whenever a hybrid automaton is either a FOCoRe or an IDA can be reduced to a decidability problem over first-order formulæ.

In the future, we plan to further study the expressiveness of first-order theories in hybrid automaton context. Since the Michael’s form can guarantee the existence of a continuous transition for any kind of first-order dynamics, we would like to investigate the possibility of relaxing it by restricting the specification theories to O-minimal theories. As a matter of fact, even if Example 4.1.3 proves the existence of a continuous map for which there is no continuous selection, such example is not O-minimal. Moreover, we are interested in the possibility of exploiting first-order theories over reals with restricted variables over naturals to study synchronisation problems over hybrid automata. Finally, we will try to apply the presented techniques to study stability of hybrid systems.

Concerning the third part of the thesis, we presented an ellipsoidal based algorithm by Botchkarev for bounded time reachability analysis, we proved that such algorithm does not always terminate, and we proposed a set of conditions which guarantees termination. We described a new method, derived from Botchkarev’s algorithm, that yielded a tighter approximation. The new algorithm avoids the approximations caused by the union operation, and so it can prove that a region is unreachable by a given hybrid automaton, also when Botchkarev’s algorithm returns otherwise. We also proved
that the two algorithms have the same asymptotic complexity. Furthermore, we introduced the notions of $\epsilon_H$-expansive automaton, approximation index of a calculus method and $(C, \delta)$-compatibility, to characterise a class of hybrid automata for which termination of the two algorithms can be proved, and we defined the algorithmic approximation index to measure the quality of reachability analysis algorithms, showing that the new algorithm has a better index than the original one. We tested the two algorithms on a real life problem, proving with the new one the correct behaviour of a controlled car engine that could not be verified with Botchkarev’s algorithm. In Chapter 8, we presented ARIADNE, an environment and open-source tool for developing algorithms for the reachability analysis of hybrid automata. We outlined the rigorous computable analysis theory on which ARIADNE is based, concentrating on the approximation of geometric objects, and on computation with provable error bounds. The proposed tool differs from existing ones in that we define a sound theoretical basis for the semantics of operators in continuous space and time, making available exact and approximate, but error-bounded operations on geometric points and sets. However, for efficiency we also allow approximate computations without known error bounds. All operations are wrapped with a well-defined interface that may be easily extended to suit the needs of different users. In this way we can treat general nonlinear hybrid systems.

In the next months, we plan to complete the ARIADNE implementation and to include in it approximate algorithms derived from both the theoretical framework proposed on Part II of this dissertation and convex approximation techniques cited in Part III. Moreover, we would like to use ARIADNE arbitrary-precision approximate representations to study both biological and engineer problems.
Bibliography


Index

[·], 26
[·]−, 26
⟨·⟩, 26

\(\mathcal{A}(\cdot), 24\)
\(\text{Act}(\cdot)[·], 24\)
activation, 24
\(\text{ActSet}, 24\)
algorithmic approximation index, 104
approximation index, 98, 104
ARIADNE, 109

automaton
multirate, 31
rectangular, 32
axiom of choice, 38

bisimilar, 20
bisimulation, 20
quotient, 21
bisimulation relation, 20

calculus method, 94
approximation index, 98
\(\chi(\cdot)[·], 84\)
\(\chi(\cdot)[·], 84\)
choise function, 38
compatibility, 98
control mode, 24
control switch, 24
correspond, to, 29
corresponding model, 27
timed, 27
untimed, 27

CTL
syntax, 14

\(\text{CTL}^*, 13\)
semantics, 13
syntax, 13
\(\text{CTL}^*_{−o}, 15\)

semantics, 16
syntax, 15

\(\text{CTL}_{−o}, 16\)
syntax, 16

\(\Delta, 11\)
destination, 9
\(\text{Dyn}(\cdot)[·], 24\)
dynamics, 24
\(\text{DynSet}, 24\)

\(\mathcal{E}, 9\)
edge, 9, 24
subsequent, 10

\(\epsilon_{H}\)-expansive, 97
\(\epsilon_{\cdot}, 95\)

\(\eta(\cdot), 82\)
\(\eta(\cdot)[·], 82\)

\(F_{\cdot}, 40\)
f-independent variables, 73
on a location, 73
on an edge, 74
first-order automata, 30

\(\text{FOCoRe}, 51\)
syntax, 51

\(\Gamma, 23\)
\(\gamma^C, 98\)

connected graph, 9
directed graph, 9
reachability, 10

\(H_{\text{inf}}, 56\)

hybrid automaton
\(\epsilon_{H}\)-expansive, 97
activation, 24
constant reset, 24
corresponding model, 27
Index

dynamics, 24
edge, 24
first-order, 30
invariant, 24
location, 24
model checking, 29
O-minimal, 34
reachability, 28
region, 25
reset, 24
semantics, 25
simple, 32
state, 25
syntax, 23
trace, 27
trajectory, 27

$I_·, 40
I(·), 24
IDA, 69
syntax, 75
improper subgraph, 70
induced partition, 73
Inv(·)[], 24
invariant, 24
InvSet, 24

j_{\text{min}(·)}, 96
ΛS, 11
LR, 4
location, 24
lower semi-continuous function, 38
LTL
syntax, 14
LTL_{ad}, 16
syntax, 16

M", 27
M', 27
M\text{Form}, 42
Michael's form, 40
MinDyn(·), 75
model, 11
arc, 11

continuous label, 11
discrete label, 11
edge, 11
dge's label, 11
finite, 11
infinite, 11
state, 11
model checking, 18
M_R, 5
M_0, 5
µ(·)[], 82
multirate automaton, 31
n-rated, 31

ν(·)[], 83

O-minimal

hybrid automaton, 34

\overline{P}, 53
P^{|·|}, 10
P, 10
partition
induced, 73
persistent, 74
path, 10
bisimilar, 20
bisimulating, 20
corresponding, 29
simulated, 19
simulating, 19
infinite, 10
persistent partition, 74
Φ_U, 65
Φ, 64
ψ, (·), 99
\tilde{ψ}, (·), 102
Q, 11
quotient
bisimulation, 21
simulation, 20

R(·), 24
reachability, 10
reachability problem, 28
Reach((.,·)[:,], 43
Reach((.,·)[:,], 46
rectangular automaton, 32
closed activity, 33
initialised, 33
uniform activity, 33
region, 25
Reset((·)[]), 24
Reset(·), 24
reset, 24
constant, 24
ResetSet, 24
ρ(·)[], 76
ρ(·), 73
ρ(·)[], 76
ϕ(·), 64
ϕ(·), 65
ruling formula, 73
on a edge, 74
on a location, 73
S, 11
S+·, 96
Sat(·), 23
selection, 38
selection problem, 38
σ(·)[], 76
σ(·), 73
ϕ(·)[], 76
ξ(·)[], 84
simulate, to, 19
simulation, 19
quotient, 20
simulation equivalence, 20
simulation relation, 19
Shv(·), 79
source, 9
state, 25
subgraph, 10
improper, 70
T., (·), 95
TCTL, 17
semantics, 17

syntax, 17
Test(·), 88
touching path component, 70
Tp(·)[:-], 42
trace, 27
trajectory, 27
transitive

dynamics, 25
formula, 25
transitivity, 25
unrelated formula, 73
on a edge, 74
on a location, 73
Υ, 104
V+, 95
vertex, 9
V, 95

Zeno
hybrid automaton, 28
trajectory, 28