Reachability Problems for Systems with Linear Dynamics

by

Shang Chen

A Doctoral Thesis

Submitted in partial fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Loughborough University

Supervised by Dr Paul Bell
Dr Lisa Jackson

21th March 2016

Abstract

This thesis deals with reachability and freeness problems for systems with linear dynamics, including hybrid systems and matrix semigroups. Hybrid systems are a type of dynamical system that exhibit both continuous and discrete dynamic behaviour. Thus they are particularly useful in modelling practical real world systems which can both flow (continuous behaviour) and jump (discrete behaviour). Decision questions for matrix semigroups have attracted a great deal of attention in both the Mathematics and Theoretical Computer Science communities. They can also be used to model applications with only discrete components.

For a computational model, the reachability problem asks whether we can reach a target point starting from an initial point, which is a natural question both in theoretical study and for real-world applications. By studying this problem and its variations, we shall prove in a formal mathematical sense that many problems are intractable or even unsolvable. Thus we know when such a problem appears in other areas like Biology, Physics or Chemistry, either the problem itself needs to be simplified, or it should by studied by approximation.

In this thesis we concentrate on a specific hybrid system model, called an HPCD, and its variations. The objective of studying this model is twofold: to obtain the most expressive system for which reachability is algorithmically solvable and to explore the simplest system for which it is impossible to solve. For the solvable sub-cases, we shall also study whether reachability is in some sense "easy" or "hard" by determining which complexity classes the problem belongs to, such as P, NP(-hard) and PSPACE(-hard). Some undecidable results for matrix semigroups are also shown, which both strengthen our knowledge of the structure of matrix semigroups, and lead to some undecidability results for other models.

Acknowledgements

I would like to especially acknowledge my main supervisor, Dr Paul C. Bell, for guiding and supporting me over the past three and one half years. He has been incredibly patient and kind and of course, could always come up with interesting research ideas. Without his help, the completion of my Ph.D would not have been possible.

I would also given a huge thanks to my second supervisor, Dr Lisa M. Jackson, for her useful suggestions which has helped me work more efficiently during my Ph.D study.

The Department of Computer Science at Loughborough University has been an excellent place to conduct research and all members of staff here have been friendly and helpful. I would also like to kindly thank the Graduate School for funding my postgraduate studies.

Last but no least, to my mum Fan Zhang and dad Weiyou Chen, an eternal gratitude for both financial and spiritual support not only over the past three and one half years, but since the beginning of my life. I would feel your love even it has been thousands miles away.

Contents

List of Figures

List of Tables

Notation Glossary

Basic Notation

$\mathbb{N}$ - The set of natural numbers ($\{0, 1, 2, \ldots\}$).

$\mathbb{Z}$ - The ring of integers.

$\mathbb{Q}$ - The field of rational numbers.

$\mathbb{F}$ - Arbitrary ring of numbers.

$f^t(x) = \underbrace{f(f(\ldots f(x)\ldots))}_{t}$ - The t times iteration of function f.

Group Theory Notation

$A \cup B$ - The union of two sets A and B.

$A \cap B$ - The intersection of two sets A and B.

$|A|$ - The cardinality of the set A.

$\langle \mathcal{G} \rangle$ - The semigroup generated by a set of square matrices $\mathcal{G}$.

$\mathcal{S}$ - A matrix semigroup.

$\Lambda(G)$ - The set of scalars generated by a set of square matrices $\mathcal{G}$ and two given vectors.

Matrix Notation

$M_{[i,j]}$ - The element in the i'th row and j'th column of matrix M.

$\mathbf{0}$ - The zero matrix with appropriate dimensions.

$M \oplus N$ - The direct sum of matrices M and N.

Word Notation

ε - The empty word.

$u \cdot v = uv = u_1u_2 \cdots u_nv_1v_2 \cdots v_m$ - The concatenation of u and v, where $u = u_1u_2 \cdots u_n$ and $v = v_1v_2 \cdots v_m$.

$|u|$ - The length of word u.

A^* - The set of all words over finite set of letters A.

A^+ - The set of nonempty words over finite set of letters A.

Chapter 1

Introduction

1.1 Background and Known results

In this thesis we shall study two types of systems with linear dynamics: linear hybrid automata and matrix semigroups. We will mainly focus on the reachability problem and some related problems like mortality and freeness on these systems. We start with an introduction to hybrid automata.

1.1.1 Hybrid Automata

Hybrid automata are an important class of mathematical model allowing one to capture both discrete and continuous dynamics in the same framework. There is currently much interest in *hybrid systems*, since they can be used to model many practical real-world systems in which we have a discrete controller acting in a continuous environment. Their analysis has a huge range of potential applications, such as aircraft traffic management systems, aircraft autopilots, automotive engine control [8], chemical plants [10] and automated traffic systems for example.

Hybrid systems are described by a state-space model given by the Cartesian product of a discrete and continuous set. The system evolves over time according to a set of defined rules, such as differential equations or differential inclusions for example, until some condition is satisfied. At this point a discrete, non-continuous event occurs. Such an event can cause an update to certain variables and change the continuous dynamics of the continuous variables. See Section 2.1.3 of Chapter 2 for formal definitions.

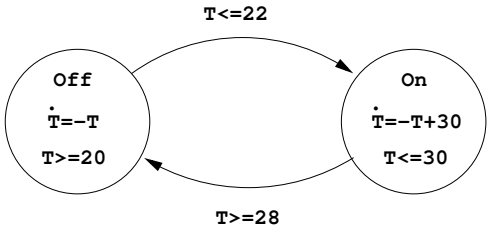


Figure 1.1: An example of Hybrid Automaton - Thermostat Control

The example in Figure 1.1 shows a thermostat control modelled by a hybrid automaton. There is a set containing two discrete states: "on" and "off", representing the states of the thermostat control being turned on or shut off, respectively. If the thermostat control is at "off" state, the temperature T will drop continuously at rate $-T$ according to the given differential equation $\dot{T}(t) = -T(t)$. The condition $T \geq 20$ is formally called an invariant, restricting that the thermostat control can be kept off only if the temperature is above 20 degrees. The condition $T \leq 22$ is formally called a guard, indicating the thermostat control can be switched to "on" state whenever the temperature in below 22 degrees. Note that when T is between 20 and 22, the system can either be in the "on" or "off" state, so it is a nondeterministic system.

As reported in [26], in the past it was common to suppress the hybrid nature of systems by converting them into purely discrete or purely continuous systems, to allow the more richly developed analysis and control techniques of these 'pure' systems to be employed. Yet the trend towards ever more complex embedded systems mean that formal techniques for analysing hybrid systems are becoming ever more crucial.

In this thesis we will concentrate on hybrid systems with linear dynamics. The reason is, hybrid systems with linear dynamics are expressive enough to be able to model a rich variety of real-world systems and even for hybrid systems with some simple dynamics, many decision problems will become undecidable. They are therefore a useful model to explore the frontiers of decidability for reachability problems. We will give examples of hybrid systems with simple dynamics later on. But before that, the reader should first understand the concept of undecidability.

In Theoretical Computer Science, some of the most important research areas are: automata theory, computability theory and complexity theory. The study

of Theoretical Computer Science often begins with automata theory, as we must know how a "computer" is formally defined. Automata theory is an area that deals with the definitions and properties of abstract "computers" which are formally called mathematical computational models. From automata theory, we know that different computational models are defined according to different purposes. For example, a well-known model in Theoretical Computer Science is that of a finite automaton. A finite automaton is defined to model devices that have a finite amount of memory and can be used in applied areas like the text processing, compilers and hardware design. Another famous model is the Turing machine, which is a very powerful model with infinite and unrestricted memory and is able to solve every problem that can be done by a real computer, according to the Church-Turing thesis.

However, even though Turing machines are extremely powerful abstract computational models (equivalent in power to real-world computers, in a formal sense), there are still problems that can not be solved by them. This is why, before considering the computational resources (such as time and space) required to solve a problem, as is carried out in *complexity theory*, we first need to answer the question: whether this problem can be solved at all? We call this area *computability theory*.

More formally speaking, by "solving" a problem we mean that for a decision problem (to which the answer is yes or no) it is possible to construct a single algorithm that always leads to a correct yes-or-no answer. If it can be proven that for a decision problem such an algorithm does not exist, then the problem is called *undecidable*; if such an algorithm does exist, then the problem is said to be *decidable*.

The reader may be surprised to know that there are problems that can not be solved by a "computer". But in fact, there are many such problems known. For instance, the undecidable problems that are used in this thesis including the *halting problem*, *generalised Collatz problem*, *Hilbert's tenth problem*, and *Post's correspondence problem*. See Section 2.1 for their definitions and more details. In computability theory, research usually goes to two aspects: exploring the most complex conditions that keep decidability and the simplest conditions that lead to

undecidability. We call this exploring the frontiers of decidability for a particular problem. By studying computability theory, if we know a problem is undecidable, then it must be simplified in some way before we can find an algorithmic solution.

When a problem is known to be solvable (formally called decidable), we can study whether it is a "hard" or "easy" problem and what properties make a problem more difficult than others. The research area that deals with such problems is called complexity theory. The problems are classified by time, memory or other computational resources required to solve them. We specify the number of required resources in terms of the input size to the problem. Usually, we consider the worst-case complexity of the problem: in other words, what is the maximum number of resources required for the problem, by any instances of some size n.

The most common complexity classes include P, NP, PSPACE and EXPTIME. Generally speaking, the classes P and NP are the problems that can be solved in polynomial time by a deterministic or nondeterministic Turing machine, respectively. The class EXPTIME are those problems that can be solved in exponential time by a deterministic Turing machine. These three classes are classified with respect to the *worst-case time complexity* of a problem. The class PSPACE is classified with respect to *worst-case memory space* ,which contains all the problems that are solvable in a polynomial amount of space on a (deterministic or nondeterministic) Turing machine. The relationships of the above four classes are P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME, and at least one of the containments is proper. Although many researchers believe that all the containments are proper, no one has proven them yet. Also note that, even though some problems may be decidable theoretically such as the problems in EXPTIME, it may be impossible to solve them in practice since even for relatively small input sizes, the amount of time required to give a solution may be prohibitively expensive. As the input size of the problem grows, the time required grows extremely fast. See [59] for a detailed discussion about automata theory, computability theory and complexity theory.

Reachability for Hybrid Systems

A fundamental question concerning hybrid systems is that of *reachability*: does there exist a trajectory starting from some initial state (or set of states) which evolves to reach a given final state (or set of states) in finite time (defined formally in Section 2.1.3, Chapter 2)? Related questions, such as *convergence* (does there exist a state, or periodic set of states, towards which the system converges for any initial state) or *control problems* (given an input, can the system be controlled to avoid some 'bad' set of states?), are also important, see [23], for example. But we will not consider them in this thesis.

Unfortunately, as mentioned above, many reachability problems are undecidable, even for very restricted hybrid systems [4, 7, 22, 39]. Studying the boundary of decidability for reachability problems on classes of hybrid systems thus allows us to define what is algorithmically achievable. The objective of studying the decidability boundary is twofold; to obtain the most expressive system for which reachability is decidable and to study the simplest system for which it is undecidable. For hybrid systems with more complex dynamics, since the decidability for which are high likely to be undecidable, people often use the method of approximation to study them, see [1] for example.

One well-known hybrid system with simple dynamics is that of a *Timed Automata* (TA). A timed automaton is a nondeterministic hybrid system such that all continuous variables have derivative 1 (so they are also called clocks), all resets are constants, guards are non-comparative (i.e. they do not contain conditions like $x > 2y$) and all invariants are constants intervals, see [3] for full details. Timed automata and their variants are widely studied and a lot of results are known. The reachability problem for timed automata is decidable [3]. However, releasing some restrictions on timed automata may lead to undecidability. For example, allowing guards of the form $x = 2y$ makes reachability problem for timed automata undecidable [3]. The reachability problem for stopwatch automata (timed automata that the rate of clocks can be either 0 or 1) is also undecidable [38].

Some complexity problems related to reachability in timed automata have also been studied. In [48] it was proven that for timed automata with one clock, reachability is NLOGSPACE-complete, and for timed automata with two clocks,

the problem is NP-hard. In [30] it was also shown that for timed automata with three clocks, reachability becomes PSPACE-complete.

Rectangular Automata (RA) are another well-known hybrid system model with simple dynamics. They can be seen as an extension model of timed automata. The only difference is that in rectangular automata, the derivatives of variables are bounded by constant *intervals* instead of being equal to 1. A rectangular automaton is called *initialised* if a variable is reset to a constant whenever its flow changes. The initialisation is a necessary condition for a decidable reachability problem. It was proven in [3] that the initialised rectangular automata can be translate into timed automata, thus the reachability problem is decidable, and is also known to be PSPACE-complete [2, 54]. However, the problem becomes undecidable for a uninitialised rectangular automaton with only one clock of rate 1 and some other constant k [38].

Hierarchical Piecewise Constant Derivative (HPCD) Systems

From the above, we can see timed automata and rectangular automata are two hybrid systems lying aside the boundary between decidability and undecidability. In this thesis, however, we will study another hybrid system with linear dynamics that is called a *Hierarchical Piecewise Constant Derivative* (HPCD) system. To understand the concept of an HPCD, we first introduce an important and intuitive model of hybrid system called a *Piecewise Constant Derivative* (PCD) system.

In a PCD, we partition the continuous state space into a finite number of nonempty regions, each of which is assigned a constant derivative defining the dynamics of a point within that region (see Section 2.1.3, Chapter 2 for full details). It was proven in [49] that reachability for PCD systems in two dimensions (2-PCD) is decidable, but for three dimensions (3-PCD), the problem becomes undecidable [4]. One of the important properties of a PCD, which leads to its reachability problem being decidable in dimension two, is that trajectories can never 'cross' each other since each region has a constant derivative assigned. It can be proven that the trajectories are either periodic, or else form an expanding or contracting spiral which can be proven using geometric arguments on the *edge-to-edge successor* function (also called the Poincaré map) of a 2-PCD.

In [6], a related model named an HPCD was introduced. An HPCD is a 2-dimensional hybrid automaton where the dynamics in each discrete location is given by a 2-PCD (formal details are given in Section 2.1.3, Chapter 2). Certain edges in the HPCD are called (transition) guards and cause the HPCD to change location if ever the trajectory reaches such an edge. When transitioning between locations, an affine reset rule may be applied. If all regions of the underlying PCDs are bounded, then the HPCD is called bounded. This model can thus be seen as an extension of a 2-PCD.

The reason why we are interested in the HPCD system not only because it is an extension of a 2-PCD and thus is an intermediate model lying between the decidable 2-PCD and the undecidable 3-PCD, but also as it links the continuous PCD model to an important simple discrete 1-dimensional system called the 1-dimensional *Piecewise Affine Map* (1-PAM). A 1-PAM is a piecewise function which is applied to the 1-dimensional real line, such that the function within each interval of the real line is affine (see Section 2.1.4 for details). Though the 1-PAM looks like a simple system, our understanding of it is quite limited. The reachability problem for 1-PAMs is stated as an open problem in [5, 19, 20, 43, 45], but it becomes undecidable in the 2-dimensional case with fewer than 800 intervals [43]. However, our knowledge of reachability for 1-PAMs is so lacking that even for a 1-PAM with only two intervals, the problem remains open to the author's knowledge.

In [45], 1-PAMs are proven to be equivalent to a 2-dimensional system called a planar pseudo-billiard system, also known as a "strange billiards" model in bifurcation and chaos theory [56] (see '*simulations*' under Section 2.1.3 for the definitions of equivalence and simulation). Some decidable results are known under restricted cases. In [5], it is proven that reachability is decidable for 1-dimensional Onto PAMs, which is a model such that every interval in the PAM can be exactly mapped to another. In [19], it is shown that for 1-PAMs over the integers (where all coefficients, the initial point and the final point are integers), the reachability problem is PSPACE-complete, which implies that reachability for rational 1-PAMs is at least PSPACE-hard. If PAMs are replaced by polynomials, the decidability of the reachability problem is open for any dimension [6]. If PAMs are replaced

by piecewise rational maps, the reachability problem is undecidable even for dimension one [45]. Point to set reachability for two interval PAMs is considered in [17]. The density of orbits in PAMs is studied in [46].

In the above we mentioned that the HPCDs link the PCDs to 1-PAMs because reachability for bounded 1-PAMs was shown to be equivalent to that of reachability for bounded HPCDs with either: i) comparative guards, identity resets and elementary flows in Proposition 3.20 of [5] or else ii) affine resets, non-comparative guards and elementary flows in Lemma 3.4 of [5] (See Section 2.1.3 for definitions).

Related to the reachability problem is the *mortality problem*. The mortality problem is the problem of determining if the trajectories starting from *all* initial points/configurations eventually halt (defined formally in Section 2). The mortality problem has been studied in many different contexts [15, 19, 20, 24, 35] and has connections with program verification, especially in a discrete setting. Similar to the case of reachability, the mortality for 1-PAMs is also stated as an open problem in [19, 20], and undecidability also starts at dimension two, in both the integer case [19], and for the rational case [20]. Global convergence is also known to be undecidable in dimension two [20], although both problems are decidable in dimension one when the piecewise affine function is continuous. The author of [19] also shows Π_2^0-completeness for the integer case.

However, neither reachability nor mortality is a superclass of the other. For the mortality problem, we must prove that *all* initial points will eventually halt, or else the system can be called *immortal* (meaning that the system may diverge, become periodic or quasi-periodic for example). Mortality for 1-PAMs over the integers is known to be PSPACE-complete [19]. Whether the undecidability results in dimension two still hold for a fixed number of intervals is unknown, in both the rational and integer cases.

Similarly to [5], we also aim to study the following question: "What is the simplest class of hybrid systems for which reachability is intractable or undecidable?" To this end, we define the model of *Restricted HPCD* (RHPCD), which is a deterministic bounded HPCD with (1) elementary flows (derivatives of all continuous variables come from $\{0, \pm 1\}$), (2) identity resets and (3) non-comparative guards and is thus a simpler form of HPCD. Certain power subset of these re-

strictions on HPCDs have been considered in [5]. These restrictions on the resets, derivatives and guards seem natural ones to consider. For example, restricting to identity resets means the trajectory will not have discontinuities in the continuous component, which is similar to a PCD trajectory. Restricting the derivatives to elementary flows ($\{0, \pm 1\}$) has similarities to a *stopwatch automaton*, for which all derivatives are from $\{0, 1\}$. Restricting the guards to be non-comparative gives strong similarities to the guards of a *rectangular automaton* [38], as well as the diagonal-free clock constraints of an *updatable timed automaton* [25].

	∞ num. of regions	Linear resets	Affine resets	Comparative guards	Arbitrary const. flows	Num. of locations	
Decidable	×	×	×	×	×	$N < \infty$	*
	×	×	×	✓	✓	1	[49]
1-PAM equivalent	×	×	×	×	✓	$\lceil \log_2 n \rceil + 3$	*
	×	×	×	✓	×	$4n$	[5]
	×	×	✓	×	×	1	[5]
	×	✓	×	×	×	$\lceil \log_2 n \rceil + 3$	*
Undecidable	✓	×	×	×	×	1	[5]

Table 1.1: Summary of decidability status of the reachability problem for 2-RHPCDs when certain conditions are allowed (✓) or disallowed (×). Starred results are contributions of this thesis.

We prove that a bounded 1-PAM can also be simulated by an RHPCD with arbitrary constant flows or with *linear* resets. Together with the results in [5], the reachability problem for bounded HPCDs is thus shown to be equivalent to that of bounded 1-PAMs when the HPCD only has one of the following: comparative guards, linear resets (or affine resets) or arbitrary constant flows, see Table 1.1 for an overview.

We then consider an n-dimensional analogue of RHPCDs, which we denote n-RHPCDs. We show that reachability is decidable (and in PSPACE) for bounded n-RHPCDs and mortality is decidable for bounded 2-RHPCDs. We show a lower bound that reachability and mortality for bounded 3-RHPCDs is co-NP-hard.

We also extend the n-RHPCD model with nondeterminism and unboundedness. If the 2-RHPCD model is endowed with a nondeterministic transition function between locations, then the reachability problem becomes PSPACE-hard. Furthermore, we show that the reachability and mortality problems for unbounded 3-RHPCDs is actually undecidable by an encoding of a Minsky machine.

Note that the reachability problem for a 3-HPCD is undecidable, even with only one location, since HPCDs are a superclass of 3-PCDs for which reachability is undecidable [4].

1.1.2 Matrix Semigroups

Matrices are one of the fundamental objects in Mathematics. They are used in a wide variety of areas of mathematics, such as representing linear equations, graphs, Markov chains and probability distributions. Matrices are also crucial for real-world applications, for instance computer graphics; a world without matrices would be a world without video games.

In this thesis we concentrate on matrix semigroups, which are semigroups generated by a finite set of square matrices, together with the operation of matrix multiplication (see Section 2.1.2 for formal definition). Similar to the hybrid system part, we will also study reachability type problems for matrix semigroups.

Firstly, we should mention one of the most basic and natural problems for matrix semigroups - the *membership problem*, which is to determine whether a given element (a matrix) M is contained within a matrix semigroup $\mathcal{S}$. Reachability type problems for matrix semigroups are related to, or can be regarded as the variation of the membership problem. The *vector reachability problem* asks given two vectors x, y and a matrix semigroup $\mathcal{S}$, whether exists a matrix $M \in \mathcal{G}$ such that $Mx = y$. The *scalar reachability problem* is the question to determine given two vectors x, y, a matrix semigroup $\mathcal{S}$ and a value k, whether there exists a matrix $M \in \mathcal{S}$ such that $x^T My = k$. The *mortality* and *identity problems* ask if the zero matrix or identity matrix belong to the matrix semigroup, respectively (in fact, they are special cases of the membership problem).

The *freeness problem* is to determine whether a semigroup generated by a finite set of matrices is free, or in other words, if every matrix in the semigroup has a unique factorisation over elements of the generator. The *vector ambiguity problem* asks given a vector x and a matrix semigroup $\mathcal{S}$, if it is true that $Mx = Nx$ leads to $M = N$ for all $M, N \in \mathcal{S}$. Besides, we can ask these questions over bounded languages, i.e., instead of generating a matrix M arbitrarily, we require M to be generated by a fixed order of elements of the generator (see section 2.2.2 for full

details).

A lot of research has been done and many results are known for reachability type problems for matrix semigroups. Undecidability generally starts from dimension three or four over rational numbers, for example, the membership problem (including the identity and mortality problem), vector reachability problem, scalar reachability problem and freeness problem [11, 18, 24, 37]. Dimension two is the challenging part and many problems remain open. The known results are usually based on a different semi-ring, for example, for 2×2 matrices, the membership and vector reachability problems are undecidable over quaternions [18], and the identity problem is decidable over the integers [28].

In this thesis we will introduce two new reachability type problems in matrix semigroups called *Scalar Ambiguity* and *Scalar Freeness* problems. These are related to the uniqueness of factorizations of a set of scalar values of the form $\{\rho^T M \tau | M \in \mathcal{S}\}$, where $\mathcal{S}$ is a finitely generated matrix semigroup (see Section 2.2.2 for details). We show that these two problems are also undecidable both in general case and over bounded languages, by reductions from Post's correspondence problem and Hilbert's tenth problem.

In Chapter 4, we also study a related ambiguity problem for *Probabilistic Finite Automata* (PFA), defined in Section 2.1.4. The reachability problem for PFA (or emptiness problem) is known to be undecidable [55], even in a fixed dimension [21, 42]. The reachability problem for PFA defined on a bounded language (i.e. where input words are from a bounded language which is given as part of the input), was also shown to be undecidable [16].

Associated with each input word is the probability of that word being accepted by the PFA. In this thesis, we show that determining whether every probability is unique is undecidable over a bounded language. In other words, to determine if there exists two input words which have the same probability of being accepted is undecidable. This is a similar concept to the *threshold isolation problem* shown in [21] to be undecidable, where we ask if each probability can be approximated arbitrarily closely.

1.2 Overview of the Thesis

We now illustrate how the thesis is organised. In Chapter 2, "Preliminaries", we first briefly talk about two general type problems we will look at in this thesis, the decidability and complexity problems. The proof technique of "reducibility" is also introduced, as it is used throughout this thesis. We then try to give all the formal definitions needed in this thesis for both hybrid systems and matrix semigroups.

In the hybrid systems part, we introduce the concept of "simulation", as it is a crucial method used in Chapter 3 to show some reachability results and it is important to carefully define what it means for one computational model to "simulate" another. We also list some known undecidable or complexity results, like generalized Collatz problem, Hilbert's tenth problem and simultaneous incongruences, so they can be used to prove our undecidable or hardness results by reduction or simulation. Finally we define the reachability type problems formally for both hybrid systems and matrix semigroups, together with examples to aid understanding.

Chapter 3, "*Reachability Problem for HPCDs*" deals with the reachability problem for HPCDs and its variations. We start with the two-dimensional case, summarise the computational powers of 2-HPCD formally and extend the results in [5]. This result and other relative results were presented in [12]:

- P. C. Bell, S. Chen, Reachability problems for hierarchical piecewise constant derivative systems, in: Reachability Problems, Vol. 8169 of Lecture Notes in Computer Science, 2013, pp. 46-58.

We then study the n-dimensional but restricted version of HPCD system called n-RHPCD. We are able to show the computational power of such a system is quite limited and the reachability problem for it is decidable. Hence we are more interested in complexity results. We show a lower bound for the 3-dimensional case and a upper bound for the general case. Also, we have a discussion about 1-PAM and a more general model of *Piecewise Rational Maps* 1-PRM, not only because 1-PAM has a close relation to 2-HPCD, but also 1-PAM and 1-PRM themselves are interesting models. At the end of this chapter, we show some results for RHPCD

with extensions. Some of the results mentioned above were presented in [13]:

- P. C. Bell, S. Chen, L. M. Jackson, Reachability and mortality problems for restricted hierarchical piecewise constant derivatives, in: Reachability Problems'14, Vol. LNCS 8762, 2014, pp. 32-45.

In Chapter 4, "*Mortality Problem for HPCDs*", mortality problems for different dimensional HPCDs are studied. In contrast to reachability problem, the mortality problem deals with the behaviour of trajectories starting with all valid configurations of HPCDs. We show a lower bound of the 2-dimensional case, a upper bound of a 3-dimensional case and also the unbounded case. The results shown in this chapter were also published in the above paper [13].

In Chapter 5,"Scalar Ambiguity and Freeness Problems", we introduce two new reachability type problems for matrix semigroups named *scalar ambiguity* and *scalar freeness* and show that they are undecidable, both in the general case and over bounded languages. The proof for the general case is shown by reducing from a variation of Post's correspondence problem called *Mixed Modification PCP* (MMPCP) (defined in Section 2.1.4), and the undecidability starts from dimension three and four, just like many other problems for matrices. For the problems over bounded languages, we prove the undecidablility results by an encoding of Hilbert's tenth problem, which is related to finding zeros of Diophantine equations. Reductions using Hilbert's tenth problem often require higher dimensions. Later in this chapter we also use the similar proof technique to study an ambiguity problem for PFA. The results in this chapter were presented in [14]:

- P. C. Bell, S. Chen, L. M. Jackson, Scalar ambiguity and freeness in matrix semigroups over bounded languages, in: Language and Automata Theory and Applications: 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings, Vol. 9618, Springer, 2016, p. 493.

Most of the results in this thesis were presented at 7th International workshop on Reachability Problems (RP2013), 8th International workshop on Reachability Problems (RP2014) and Language and Automata Theory and Applications (LATA2016).

Chapter 2

Preliminaries

In this chapter we outline all the concepts and problems that will appear in this thesis. They are either given by full definitions for completeness or refer to the literature.

We first briefly introduce the fundamental concept of decidability in computability and then explain how decidable problems are classified in complexity theory, as all results shown in this thesis will be decidability and complexity results. We also illustrate how the proof technique of reducibility works, as it will be used throughout this thesis.

We then list all the specific definitions and divide them into three parts: the *"Algebra, Group, Matrix and Words"* part, which includes some common definitions in the areas of algebra, matrix, groups and words; the *"Hybrid Systems"* part, which includes the definition of general hybrid automata and some specific models such as PCDs, HPCDs and RHPCDs; and the *"Discrete Computational Models"* part, which includes the definitions of some well-known discrete models that are related to this thesis, such as 1-PAM, generalised Collatz function and Minsky machine, etc, as well as some known results for them.

Finally we shall formally define some reachability type problems that will be studied in this thesis both for hybrid systems with linear dynamics and for matrix semigroups. These problems include the reachability problem, the mortality problem, the scalar ambiguity problem and the scalar freeness problem.

2.1 Definitions

In this thesis we use $\mathbb{R}, \mathbb{Q}, \mathbb{Z}, \mathbb{N}$ to denote the set of real numbers, rational numbers, integer numbers and natural numbers, respectively. Also define *big-O* notation $f(n) = O(g(n))$ if there exist positive integers c and n_0 such that for functions f and g, $f(n) \leq cg(n)$ for every integer $n \geq n_0$.

2.1.1 Computability and Complexity

A *decision* problem performs a calculation of an instance of a problem on a computational device (such as Turing machine) and always returns a "yes" or "no" answer.

A *Turing machine* is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where

(1) Q is the finite set of states,

(2) Σ is the finite set of input alphabet not containing the special *blank* symbol $\square$,

(3) Γ is the finite set of tape alphabet where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$,

(4) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,

(5) $q_0 \in Q$ is the initial state,

(6) q_a is the accept state,

(7) q_r is the reject state where $q_a \neq q_r$.

A Turing machine M starts with some finite input word $w \in \Sigma^*$ written onto the tape, in the initial state $q_0 \in Q$ and with its 'read/write head' positioned at the left hand end of the tape on the first cell. At each discrete time step, the machine uses the transition function δ to determine what to do, based on its current state, and the current symbol that the read/write head is on. Given a rule such as $\delta(q_i, a) = (q_j, b, R)$ for example, means that if we are in state $q_i \in Q$, reading symbol 'a' $\in \Gamma$, then the machine will change to state q_j, replace the symbol 'a' with 'b' at the current tape position, and then move the read/write head one position to the right (R).

If M reaches the accept state q_a, then machine M is said to *accept* word w and the computation *halts*. If M ever reaches the reject state q_r, then the machine M is said to *reject* word w and the computation halts. We may assume that the machine cannot move left of position 1 of the tape, but the tape to the right of the initial input word is infinite (with blank symbols).

The Turing machine we defined here is single-tape and deterministic which means it operates on a single tape, and for each $(q_i, c) \in Q \times \Gamma$, there exists at most one $(q_j, c', D) \in Q \times \Gamma \times \{L, R\}$ such that $\delta(q_i, c) = (q_j, c', D)$, thus it always performs the same computation for the same given input. See a full explanation and variants of Turing machines (multi-tape, nondeterministic) in [59]. A Turing machine is *nondeterministic* if the transition function above is of the form $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$, where $\mathcal{P}$ denotes the power set, and there are more than one (q_j, c', D) satisfy $\delta(q_i, c) = (q_j, c', D)$ for at least one (q_i, c). These variants have the same computational power as the one we defined here, and they all can solve every problem that can be done by a real computer according to the Church-Turing thesis.

A decision problem is said to be *decidable* if it is possible to construct a single algorithm such that after some finite amount of time it always returns the correct "yes" or "no" answer to the problem for any given legitimate input. If it can be proven that such an algorithm does not exist, the problem is called *undecidable*.

For decidable problems, we distinguish them by different complexity classes. It is not easy to formally define these classes in a few words, we only give a brief introduction here, see [59] for details. In the following definitions, we always consider the *worst-case* time complexity of the problems, for example the maximum amount of time or space required by an algorithm when taken over all inputs of some size n. We first need the following definitions:

A *language of a machine* M (for example, finite automaton, Turing machine) is the set of all strings accepted by M (also see Section 2.1.2).

Let $t : \mathbb{N} \to \mathbb{N}$ be a function. Define the time complexity class $\text{TIME}(t(n))$ to be $\text{TIME}(t(n)) = \{L | L \text{ is a language decided by an } O(t(n)) \text{ time Turing machine}\}$, where an $O(t(n))$ time Turing machine is a Turing machine that halts on all inputs and has running time at most $O(t(n))$.

Now we can define the class P as $\mathrm{P} = \bigcup_{k\in\mathbb{N}} \mathrm{TIME}(n^k)$. In other words, a problem is in the class P if it can be solved by a deterministic algorithm that runs in polynomial time.

Let $\mathrm{NTIME}(t(n)) = \{L|L$ is a language decided by an $O(t(n))$ time non-deterministic Turing machine$\}$. We define the class NP as $\mathrm{NP} = \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k)$. The class NP captures those problems for which it may be difficult to *find* a solution, but for which *verifying* the solution is reasonably efficient.

A problem is in the class co-NP if its complement is in the class NP. Just as NP can be considered to be the class of problems with a succinct "yes" verifier, co-NP can be considered to be the class of problems with a succinct "no" verifier.

Let $t : \mathbb{N} \to \mathbb{N}$ be a function. Define *space complexity classes* $\mathrm{SPACE}(f(n))$ to be $\mathrm{SPACE}(f(n)) = \{L|L$ is a language decided by an $O(f(n))$ space Turing machine$\}$, where an $O(f(n))$ space Turing machine halts on all inputs and scans a maximum number $O(f(n))$ of tape cells on any input of length n.

We define the class PSPACE as $\mathrm{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathrm{SPACE}(n^k)$. In other words, a problem is in the class PSPACE if it can be solved by a deterministic algorithm that uses an amount of space which is polynomial in the size of its input. It is not difficult to see that $\mathrm{P} \subseteq \mathrm{PSPACE}$, because any machine that runs in time T uses at most T space since at most one new memory cell can be visited at each step of computation. Also, we have $\mathrm{NP} \subseteq \mathrm{PSPACE}$ since $\mathrm{NP} \subseteq \mathrm{NPSPACE}$ for a similar reason and NPSPACE = PSPACE, but we omit the definition of NSPACE here, see [59] for more information.

A decision problem Q is said to be *complete* for a set of decision problems $\mathcal{S}$ if: (i) Q is a member of $\mathcal{S}$ and (ii) every problem in $\mathcal{S}$ can be reduced to Q in polynomial time (see the definition of reducibility in the following part). For example, if a problem Q is PSPACE-complete, then Q must be in PSPACE and every problem Q' in PSPACE is polynomial time reducible to Q. If Q merely satisfies condition (ii), we say that Q is PSPACE-hard.

In this thesis, we will show a number of computability and complexity results. The primary method that is widely used in proving such problems is called *reducibility*. A *reduction* is a way of converting problem A into problem B in such a way that a solution to problem B can be used to solve problem A. We say problem

A is reducible to problem B, if there exists a reduction from A to B, i.e., we can solve A by a solution to B. An intuitive example can be reducing the problem of solving a system of linear equations to the problems of inverting a matrix.

If a problem A is reducible to problem B in an efficient way (for example, in a polynomial time or space, see the following paragraph), then we know B is at least as "hard" as A. In computability theory, the "hardness" can refer to decidability. If we know A is undecidable and A is reducible to B, then B must also be undecidable (otherwise the solution to B can also solve A which gives a contradiction). It is also clearly that for undecidable problems the reduction is transitive. Some well-known examples include the first known undecidable problem, halting problem for Turing machine, is reducible to other problems like the halting problems for Minsky machine, Post's correspondence problem, etc (defined later in this chapter). The later problems are more often used in undecidable proof due to their properties.

In complexity theory, the "hardness" can refer to the complexity class the problem belongs to. For instance, if A is known to be in class NP-hard and A is polynomial time reducible to B, then B is at least NP-hard. We do not formally define "polynomially reducible " here, readers can understand this as saying that the description size of A and B are almost the same (up to a polynomial difference). There are many different examples of proofs shown by reducibility, see [33] for a thorough discussion.

2.1.2 Algebra, Groups, Matrices and Words

In this section we briefly introduce some basic concepts in algebra, group theory, matrix theory and words. Though the definitions are mostly related to Chapter 5, some of them are very basic and thus also needed in Section 2.1.3 and Section 2.1.4.

A *set* is a collection of distinct objects (called the *elements* of a set). Given two sets A and B, the *union* of A and B, denoted by $A \cup B$, is the set of all elements that either belong to A or B. The *intersection* of A and B, denoted by $A \cap B$, is the set of all elements that both belong to A and B. The *relative complement* of B in A, denoted by $A \setminus B$, is the set of all elements that belong to A but do not belong to B. For example, let $A = \{a, b\}, B = \{b, c\}$, then

$A \cup B = \{a, b, c\}, A \cap B = \{b\}, A \setminus B = \{a\}$.

A *semigroup* $(\mathcal{S}, \circ)$ is a set $\mathcal{S}$ together with a binary operation $\circ$ that satisfies the associative property - $(a \circ b) \circ c = a \circ (b \circ c)$, such that if $a, b \in \mathcal{S}$, then $a \circ b \in \mathcal{S}$ holds. It is a standard abuse of notation to refer to the semigroup itself by $\mathcal{S}$.

A *semigroup homomorphism* is defined as a mapping $\gamma : A \to B$ between two semigroups A and B if the equation

$$\gamma(a \diamond b) = \gamma(a) \circ \gamma(b)$$

holds for all $a, b \in A$, where $\diamond$ is the binary operator of A and $\circ$ is the binary operator of B. In this thesis we will simply call this a homomorphism or morphism, unless otherwise stated.

We denote an $m \times n$ matrix M over a semi-ring $\mathbb{F}$ by $M \in \mathbb{F}^{m \times n}$. We use $M_{[i,j]}$ to denote the element in the i'th row and j'th column of M. We use I and $\mathbf{0}$ to denote the identity matrix and zero matrix with appropriate dimensions, respectively, i.e.,

$$I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \mathbf{0} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix},$$

if they do not cause any confusion. Given square matrices $M \in \mathbb{F}^{m \times m}$ and $N \in \mathbb{F}^{n \times n}$, we define the *direct sum* $M \oplus N$ of M and N by:

$$M \oplus N = \begin{pmatrix} M & \mathbf{0} \\ \mathbf{0} & N \end{pmatrix}.$$

Given a finite set of matrices $\mathcal{G} \subseteq \mathbb{F}^{n \times n}$, $\langle \mathcal{G} \rangle$ is the semigroup generated by $\mathcal{G}$.

Let $A = \{x_1, x_2, \ldots, x_k\}$ be a finite set of *letters* called an *alphabet*. A word w is a finite sequence of letters from A, the set of all words over A is denoted A^* and the set of nonempty words is denoted A^+. The *empty word* is denoted by ε. We use $|u|$ to denote the length of a word u, i.e. how many letters the word u contains. Also we have $|\varepsilon| = 0$. For two words $u = u_1 u_2 \cdots u_i$ and $v = v_1 v_2 \cdots v_j$,

where $u, v \in A^*$, the concatenation of u and v is denoted by $u \cdot v$ (or by uv for brevity) such that

$$u \cdot v = u_1 u_2 \cdots u_i v_1 v_2 \cdots v_j.$$

Given a word $u = u_1 u_2 \cdots u_i$, a *prefix* of u is any word $u = u_1 u_2 \cdots u_j$, where $j \leq i$. A subset L of A^* is called a *language*. A language $L \subseteq A^*$ is called a *bounded language* if and only if there exist words $w_1, w_2 \ldots, w_m \in A^+$ such that

$$L \subseteq w_1^* w_2^* \cdots w_m^*.$$

For a semigroup $(\mathcal{S}, \cdot)$, where $\cdot$ denotes the operation of concatenation, and a subset $\mathcal{G}' \subseteq \mathcal{S}$, we say that $\mathcal{G}'$ is a *code* if $x_1 \cdots x_{k_1} = y_1 \cdots y_{k_2}$, where $x_i, y_i \in \mathcal{G}'$ implies that $k_1 = k_2$ and $x_i = y_i$ for $1 \leq i \leq k_1$. Alternatively stated, $\mathcal{G}'$ is not a code if and only if some element of $\mathcal{S}$ has more than one factorization over $\mathcal{G}'$. We call $\mathcal{G}'$ a *prefix code* if no $w_1 \in \mathcal{G}'$ is a prefix of another word $w_2 \in \mathcal{G}'$. Given a set $\mathcal{G} \subseteq \mathbb{F}^{n \times n}$, the *freeness problem* is to determine if $\mathcal{G}$ is a code for $\mathcal{S} = \langle \mathcal{G} \rangle$.

2.1.3 Hybrid Systems

We start this section by formally defining the most general model that is used to describe a hybrid system.

Definition 1. **[HA]** *An* n-dimensional Hybrid Automaton *(HA) [1] is a tuple* $\mathcal{H} = (\mathcal{X}, Q, f, I_0, Inv, \delta)$ *consisting of the following components:*

(1) A continuous state space $\mathcal{X} \subseteq \mathbb{R}^n$. *Each* $\boldsymbol{x} \in \mathcal{X}$ *can be written* $\boldsymbol{x} = (x_1, \ldots, x_n)$, *and we use variables* $x_1, \ldots, x_n$ *to denote components of the state vector.*

(2) A finite set of discrete locations Q.

(3) A function $f : Q \to (\mathcal{X} \to \mathbb{R}^n)$, *which assigns a continuous vector field on* $\mathcal{X}$ *to each location. In location* $l \in Q$, *the evolution of the continuous variables is governed by the differential equation* $\dot{\boldsymbol{x}} = f_l(\boldsymbol{x})$. *The differential equation is called the* dynamics *of location* l.

(4) An initial condition $I_0 : Q \to 2^{\mathcal{X}}$ *assigning initial values to variables in each location.*

(5) An invariant *Inv:* $Q \to 2^{\mathcal{X}}$. *For each* $l \in Q$, *the continuous variables must satisfy the condition Inv(l) in order to remain in location* l, *otherwise it must make a discrete transition or halt.*

(6) A set of discrete transitions δ. *Every* $tr \in \delta$ *is of the form* $tr = (l, g, \gamma, l')$, *where* $l, l' \in Q$, $g \subset \mathcal{X}$ *is called the* guard, *defining when the discrete transition can occur,* $\gamma \subset \mathcal{X} \times \mathcal{X}$ *is called the* reset relation *applied after the transition from* l *to* l'.

An HA is *deterministic* if it has at most one solution for its differential equation in each location and the guards of all the outgoing discrete transitions for each location are mutually exclusive (i.e. the intersection of any two such guards is empty). We consider deterministic HAs, unless otherwise stated. The *size* of an HA is its description size, i.e. the amount of space required to store a description of the HA under a reasonable encoding scheme (for example storing elements of $\mathbb{R}^n$ using a binary encoding). A configuration of an HA is a pair from $Q \times \mathcal{X}$. A *trajectory* of a hybrid automaton $\mathcal{H}$ over a time interval $[0, T]$ and starting from configuration $(l_0, \mathbf{x_0})$ where $l_0 \in Q, \mathbf{x_0} \in \mathcal{X}$ is a pair of functions $\pi_{l_0,\mathbf{x_0}} = (\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ such that there exists a sequence of times $t_0 = 0 < t_1 < t_2 < \ldots < t_k = T$ and

(1) $\lambda_{l_0,\mathbf{x_0}}(t) : [0, T) \to Q$ is a piecewise function constant on every interval $[t_i, t_{i+1})$.

(2) $\xi_{l_0,\mathbf{x_0}}(t) : [0, T) \to \mathbb{R}^n$ is a piecewise differentiable function and in each piece $\xi_{l_0,\mathbf{x_0}}$ is càdlàg (right continuous with left limits everywhere).

(3) $\xi_{l_0,\mathbf{x_0}}(t) \in \text{Inv}(\lambda_{l_0,\mathbf{x_0}}(t_i))$ for all $t < T$, where $t \in [t_i, t_{i+1})$.

(4) On any interval $[t_i, t_{i+1})$ where $\lambda_{l_0,\mathbf{x_0}}$ is constant and $\xi_{l_0,\mathbf{x_0}}$ is continuous,

$$\xi_{l_0,\mathbf{x_0}}(t) = \xi_{l_0,\mathbf{x_0}}(t_i) + \int_{t_i}^{t} f_{\lambda_{l_0,\mathbf{x_0}}(t_i)}(\xi_{l_0,\mathbf{x_0}}(\tau))d\tau$$

for all $t \in [t_i, t_{i+1})$.

(5) For any t_i, there exists a transition $(l, g, \gamma, l') \in \delta$ such that

(i) $\lambda_{l_0,\mathbf{x_0}}(t_i) = l$ and $\lambda_{l_0,\mathbf{x_0}}(t_{i+1}) = l'$;

(ii) $\xi^-_{l_0,\mathbf{x_0}}(t_{i+1}) \in g$ where $\xi^-_{l_0,\mathbf{x_0}}(t)$ means the left limit of $\xi_{l_0,\mathbf{x_0}}$ at t;

(iii) $(\xi^-_{l_0,\mathbf{x_0}}(t_{i+1}), \xi_{l_0,\mathbf{x_0}}(t_{i+1})) \in \gamma$.

If $(\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ is defined over $[0,\infty)$, then the trajectory is called infinite. Given a trajectory $\pi_{l_0,\mathbf{x_0}} = (\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ with sequence of times $t_0 = 0 < t_1 < t_2 < \ldots < t_k = T$, we denote by $\lambda_{l_0,\mathbf{x_0}}(t_0), \lambda_{l_0,\mathbf{x_0}}(t_1), \ldots, \lambda_{l_0,\mathbf{x_0}}(t_k)$ the *symbolic dynamics* of the trajectory, which will be unique for a deterministic HA (and can be infinite). This gives the sequence of locations that the HA visits during the trajectory from time 0 to T.

In the following we define a specific model of hybrid system, and it is a natural and intuitive model in two and three dimensions. Later on we will define and study a model that can be seen as its extension.

Definition 2. [**n-PCD**] *An* n-dimensional Piecewise Constant Derivative *(n-PCD) system [4] is a pair* $\mathcal{H} = (\mathbb{P}, \mathbb{F})$ *such that:*

(1) $\mathbb{P} = \{P_s\}_{1 \le s \le k}$ *is a finite family of nonoverlapping polytopes in* $\mathbb{R}^n$ *with nonempty interiors, where each* $P_s \subseteq \mathbb{R}^n$ *is defined as the intersection of finitely many open or closed halfspaces. We also call* P_s *a region.*

(2) $\mathbb{F} = \{\boldsymbol{c}_s\}_{1 \le s \le k}$ *is a family of vectors in* $\mathbb{R}^n$.

(3) The dynamics are given by $\dot{\boldsymbol{x}} = \boldsymbol{c}_s$ *for* $\boldsymbol{x} \in P_s$.

An n-PCD $\mathcal{H} = (\mathbb{P}, \mathbb{F})$ can equivalently be defined as a restricted type of HA which has n continuous variables, for which there is a location for each $P_s \in \mathbb{P}$, which has corresponding invariant P_s and all derivatives are constant in each location. The guards correspond to the boundary edges between polytopes and no reset is allowed during a transition. We thus see that a PCD is a partitioning of $\mathbb{P}$ into finitely many regions, each of which has an assigned constant derivative or slope. The trajectories are therefore broken lines, with breakpoints at the boundaries of regions. Points along the trajectory follow the derivative of the region they lie inside.

An n-PCD is called *bounded* if for its regions $\mathbb{P} = \{P_s\}_{1 \le s \le k}$, there exists $r \in \mathbb{Q}^+$, such that for all P_s, we have that $P_s \subseteq B_{\mathbf{0}}(r)$, where $B_{\mathbf{0}}(r)$ is an origin-

centered open ball of radius r of appropriate dimension. We define the *support set* of a PCD $\mathcal{H}$ as $\text{Supp}_{\text{PCD}}(\mathcal{H}) = \bigcup_{1 \leq s \leq k} P_s$.

In the following we slightly modify the definition of HPCD [5] to allow different dimensions to be studied.

Definition 3. [n**-HPCD**] *An n-dimensional* Hierarchical Piecewise Constant Derivative *(n-HPCD) system is a hybrid automaton $\mathcal{H} = (\mathcal{X}, Q, f, I_0, Inv, \delta)$ such that Q and I_0 are defined as in Definition 1, with the dynamics at each $l \in Q$ given by an n-PCD and for each transition $tr = (l, g, \gamma, l')$: (1) its (transition) guard $g \subseteq \mathbb{R}^n$, defined below, is a convex region of dimension $(n-1)$; and (2) the reset relation γ is an affine function of the form: $\boldsymbol{x}' = \gamma(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$, where $A \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$.*

We denote the *internal guards* of an HPCD location to be the boundary edges of the underlying PCD regions which can cause a change of dynamics when they are reached. The *transition guards* are the guards used in transitions between locations. The Invariant (Inv) for a location l is defined to be $\text{Supp}_{\text{PCD}}(l)$, minus the transition guard for that location, where $\text{Supp}_{\text{PCD}}(l)$ is the support set of the underlying PCD on l. If all the PCDs are bounded, then the n-HPCD is said to be bounded.

It can thus be seen that n-HPCDs are in fact n-dimensional linear hybrid automata. The definition of 2-HPCD, as described by [5], is given to emphasise the fact that the trajectory of a 2-HPCD "mostly behaves likely a PCD, with a few reset induced discontinuities". Therefore, the definitions of trajectories, symbolic dynamics and the reachability/mortality problems (defined in Section 2.2.1), can also be defined on HPCD and can be understood in terms of the representation as a two-dimensional linear Hybrid Automaton.

Note that we should avoid the case shown in Figure 2.1 when we define an HPCD system. We denote the interval by I which is the dot line segment in the example. The guard of location P is $g_1 = I$, jump to location Q and the guard of location Q is $g_2 = I$, jump to location P. Thus we have

$$\text{Inv}(P) = \text{Supp}_{\text{PCD}}(P) - I;$$
$$\text{Inv}(Q) = \text{Supp}_{\text{PCD}}(Q) - I.$$

Clearly, interval I is contained in neither invariant of P nor invariant of Q. So a trajectory cannot take a transition from P to Q or the other way around as it is not right continuous (see definition of trajectory above).

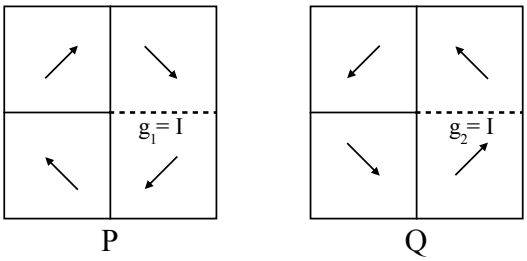


Figure 2.1: An example of an invalid HPCD

In this thesis, we are interested in a *restricted* form of n-HPCD. We first define three restrictions on an n-HPCD:

1. Under the HPCD model, when transitioning between locations, we may apply an *affine reset* to non-continuously modify the current point. An n-HPCD has identity (or no) resets if for every transition $tr = (l, g, \gamma, l')$, $\gamma(\mathbf{x}) = \mathbf{x}$ for all points $\mathbf{x} \in \mathbb{R}^n$. This means that starting from any initial configuration $(l_0, \mathbf{x_0})$, for the trajectory $\pi_{l_0,\mathbf{x_0}} = (\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ we have that $\xi_{l_0,\mathbf{x_0}}(t)$ is a continuous function of t. Note that the trajectory for a PCD is also continuous, and thus this seems to be a natural restriction.

2. An n-HPCD system has *elementary flows* if the derivatives of all variables in each location are from $\{0, \pm 1\}$, otherwise it has *arbitrary constant flows*.

3. Guards are used to change the derivative being applied within a location (internal guards), or to change which location we are in (transition guards) and can be described by Boolean combinations of atomic formulae (linear inequalities). If each atomic formula contains only one variable, then the guard is called non-comparative (meaning the guard is aligned with ones of the axes). An n-HPCD has *non-comparative guards* if all guards (both internal and transition) are non-comparative, e.g., for a 3-RHPCD, $\frac{3}{2} \leq x \leq 7 \wedge y = -1 \wedge 2 \leq z \leq 7$ is a non-comparative guard, but $0 \leq x \leq 1 \wedge 0 \leq y \leq \frac{1}{2} \wedge z = 5 \wedge x = 2y$ is a comparative guard (due to the term $x = 2y$).

Under these restrictions we can now define our restricted version of n-HPCD.

Definition 4. [n-**RHPCD**] *An* n-dimensional Restricted Hierarchical Piecewise Constant Derivative System (n-RHPCD) *is a bounded* n*-HPCD with identity resets, non-comparative guards and elementary flows. See Figure 3.6a and Figure 3.6b for an example of a* 3*-RHPCD.*

In Section 3.1 we extend the results of [5] regarding simulations of 1-PAMs by 2-HPCDs. We follow the similar approach for the definition of simulation used in [4, 5]. We define a simulation with respect to reachability. This means that if a model $\mathcal{A}$ can be simulated by a model $\mathcal{B}$, then it implies that if the reachability problem for $\mathcal{B}$ is decidable (or undecidable), then it must also be decidable (or undecidable) for $\mathcal{A}$. Since we will show simulations of both 1-PAMs and Minsky machines (defined below), we give the definition in terms of a simulation of an arbitrary *deterministic transition system*, which is a pair $\mathcal{A} = (S, \delta')$, where S is a set of states and δ' is a transition function $\delta' : S \to S$.

Definition 5. [**Simulation**] *We say that a deterministic transition system* $\mathcal{A}$*, with initial configuration* c_0 *and final configuration* c_f*, can be simulated by a* 2*-HPCD* $\mathcal{H}$ *with respect to the reachability problem if (1) configuration* c_0 *(resp.* c_f*) of* $\mathcal{A}$ *is encoded by a configuration* $(l_0, \mathbf{x_0})$ *(resp.* $(l_f, \mathbf{x_f})$*) of* $\mathcal{H}$*; (2) every configuration of* $\mathcal{A}$ *is encoded by a unique configuration of* $\mathcal{H}$*; (3) a one-step computation of* $\mathcal{A}$ *given by* $\delta'(q_k) = q_{k'}$ *is represented by a trajectory segment from* $(\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ *to* $(\lambda_{l_0,\mathbf{x_0}}(t'), \xi_{l_0,\mathbf{x_0}}(t'))$ *for some* $0 \leq t < t' < \infty$ *on* $\mathcal{H}$*, where* $(\lambda_{l_0,\mathbf{x_0}}(t), \xi_{l_0,\mathbf{x_0}}(t))$ *is the encoding of* q_k*,* $(\lambda_{l_0,\mathbf{x_0}}(t'), \xi_{l_0,\mathbf{x_0}}(t'))$ *is the configuration encoding* $q_{k'}$ *and* $(\lambda_{l_0,\mathbf{x_0}}(t''), \xi_{l_0,\mathbf{x_0}}(t''))$ *is not the encoding of any configuration of* $\mathcal{A}$ *for* $t < t'' < t'$*.*

2.1.4 Discrete Computational Models and Problems

The following model is the class of 1-dimensional Piecewise Affine Maps (1-PAM). Our approach in Section 3.1 follows a similar style to [5] where we show various classes where reachability is equivalent to that of a 1-PAM.

Definition 6. [**1-PAM**] *A* 1-dimensional Piecewise Affine Map *(*1*-PAM) is a function* $f : \mathbb{R} \to \mathbb{R}$ *(See Figure 3.3a for an example) such that:*

(1) Domain of f *: dom(*f*)=*$\bigcup I_i$*, where* I_i *are disjoint rational intervals.*

(2) $\exists a_i, b_i \in \mathbb{Q}$ *such that* $\forall x \in I_i,\ f(x) = a_i x + b_i$.

(3) f *is closed, i.e.,* $range(f) \subseteq dom(f)$.

A 1*-PAM is called* bounded *if none of its intervals is infinite. In the sequel we will write* 1*-PAM refer to bounded* 1*-PAM unless otherwise stated.*

We now state a problem for 1-PAMs, which looks easy at first glance, but is actually a longstanding open problem [5, 19, 20, 43, 45].

Open Problem 1. [1-PAM Reachability] *Given a* 1*-dimensional Piecewise Affine Map* f*, an initial point* $x \in \mathbb{Q}$ *and a final point* $y \in \mathbb{Q}$*, does there exist* $t \in \mathbb{N}$*, such that* $f^t(x) = y$*?* [1]

If we replace affine functions above by rational functions (a ratio of two polynomials), we can define a more general class of 1-dimensional piecewise maps.

Definition 7. [1-PRM] *A* 1-dimensional Piecewise Rational Map *(*1*-PRM) is a function* $f : \mathbb{R} \to \mathbb{R}$ *such that:*

(1) Domain of f : $dom(f) = \bigcup I_i$*, where* I_i *are disjoint rational intervals.*

(2) There exist polynomials $P_i(x), Q_i(x)$ *such that* $\forall x \in I_i,\ f(x) = \frac{P_i(x)}{Q_i(x)}$*, where* $Q_i(x) \neq 0$.

(3) f *is closed, i.e.,* $range(f) \subseteq dom(f)$.

A 1*-PRM is called* bounded *if none of its intervals is infinite. A* 1*-PRM is said to be of degree* k *if* $max\{D(P_i) - D(Q_i)\} = k$*, where* $D(P_i), D(Q_i)$ *denotes the degree of polynomials* P_i, Q_i*. In the sequel we will write* 1*-PRM refer to bounded* 1*-PRM unless otherwise stated.*

It was proven in [44, 45] that reachability for 1-PRM is undecidable. Later in Section 3.3 we will use a different method from [45] to show the same result but slightly improve the conditions required.

The following function is first defined by Conway [29].

[1] $f^t(x)$ denotes $\underbrace{f(f(\ldots f(x)\ldots))}_{t}$

Definition 8. [GCF] *A function $g : \mathbb{N}_+ \to \mathbb{N}_+$ is called a Generalised Collatz Function (GCF) if there exists a positive integer m together with a set of positive integers $\{a_i\}_{1\leq i\leq m}$ and a set of non-negative integers $\{b_i\}_{1\leq i\leq m}$, such that whenever $x \equiv i \pmod m$, then $g(x) = a_i(x-i)/m + b_i$.*

A standard representation of g is a set $\{m, a_1, \ldots, a_m, b_1, \ldots, b_m\}$.

We can introduce a type of reachability problem for GCFs below, which was shown to be undecidable. This result can help us show the undecidability of 1-PRM reachability.

Problem 1. [GCP] *Generalized Collatz Problem (GCP) is the problem of deciding, given a standard representation of $g(x)$, whether starting from an arbitrary point k, the trajectory of $g(x)$ reaches 1.*

Theorem 1. *[47] GCP is undecidable.*

We now introduce bounded 1-counter automaton, the reachability of which is known to be PSPACE-complete and hence can help in the proof of our PSPACE-complete result in Section 3.4.

Definition 9. [1-Counter Automaton] *A bounded 1-counter automaton [32, 34] is a tuple (Q, b, δ, l_0) such that*

(1) Q is a finite set of locations.

(2) $b \in \mathbb{N}$ is a global counter bound. The value of the counter cannot exceed b in any location.

(3) δ is a set of transitions. Each transition $tr \in \delta$ is of the form (l, p, g_1, g_2, l'), where $l, l' \in Q$ are locations, $p \in \{i \in \mathbb{Z} | -b \leq i \leq b\}$ specifies the value that should be added or subtracted from the counter c, and g_1 and g_2 is the lower and upper bound of the guard, respectively, where $g_1, g_2 \in \{i \in \mathbb{Z} | 0 \leq i \leq b\}$.

(4) $l_0 \in Q$ is the initial location.

A state of the bounded 1-counter automaton is of the form (l, c), where $l \in Q$ and $c \in \{i \in \mathbb{Z} | 0 \leq i \leq b\}$. We say there is a transition between (l, c) and (l', c'), if there is a transition $tr = (l, p, g_1, g_2, l') \in \delta$ such that $g_1 \leq c \leq g_2$ and $c' = c + p$.

Note that the bounded 1-counter automaton is a *nondeterministic* model, which means there may exist more than one transition between two locations, and from one location it can jump to more than one locations.

The reachability problem for bounded 1-counter automaton asks, starting from the initial state $(l_0, 0)$, whether the automaton can reach a final state (l_f, c_f).

Theorem 2 ([32]). *The reachability problem for bounded* 1*-counter automata is PSPACE-complete.*

In order to prove our undecidability result for an unbounded 3-RHPCD later in the thesis, we will require the following well-known computational model.

Definition 10. [Minsky machine] *Informally speaking, a Minsky machine is a two-counter automaton that can increment and decrement counters by one and test them for zero. It is known that a two-counter Minsky machine represents a universal model of computation [52]. Due to their simple structure, Minsky machines are often convenient for proving undecidability results.*

We can represent a counter machine as a simple imperative program $\mathcal{M}$ consisting of a sequence of instructions labelled by natural numbers from 1 *to some $L \in \mathbb{N}$. Any instruction is one of the following forms:*

l: **Add** 1 **to** c_k; **goto** l';

l: **If** $c_k \neq 0$ **then subtract** 1 **from** c_k; **goto** l';
else goto l'';

l: **Halt**;

where $k \in \{1, 2\}$ and $l, l', l'' \in \{1, \ldots, L\}$.

The machine $\mathcal{M}$ starts executing with some initial nonnegative integer values in counters c_1 and c_2 and the control at instruction labelled 1*. We assume the semantics of all above instructions is clear. Without loss of generality, one can suppose that every machine contains exactly one instruction of the form l : **Halt** which is the last one ($l = L$). It should be clear that the execution process (run) is deterministic and has no failure. Any such process is either finished by the execution of L : **Halt** instruction or lasts forever.*

As a consequence of the universality of Minsky machines, their halting problem is undecidable:

Theorem 3 ([52]). *It is undecidable whether a two-counter Minsky machine halts when both counters initially contain* 0.

In order to show the complex results In Section 3.2, Section 3.3 and Section 4.1, we will require the following *simultaneous incongruences problem*, which is known to be NP-complete [33, 60].

Problem 2. [**Simultaneous incongruences**] *Given a set* $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ *of ordered pairs of positive integers with* $a_i \leq b_i$ *for* $1 \leq i \leq n$. *Does there exist an integer* k *such that* $k \not\equiv a_i \pmod{b_i}$ *for every* $1 \leq i \leq n$?

We introduce another well-known undecidable problem.

Problem 3. [**Hilbert's Tenth Problem (HTP)**] *The following problem was stated as part of 23 open problems for the 20th century by David Hilbert in his 1900 address:*

"Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers".

To use a more modern terminology, Hilbert's tenth problem is to determine if there exists $n_1, n_2, \ldots n_k \in \mathbb{N}$ *such that* $P(n_1, n_2, \ldots, n_k) = 0$, *where* P *is a Diophantine equation (i.e.* P *is a polynomial with integer coefficients).*

The undecidability of Hilbert's tenth problem was shown in 1970 by Yu. Matiyasevich building upon earlier work of many mathematicians, including M. Davis, H. Putman and J. Robinson. For more details of the history of the problem as well as the full proof of its undecidability, see the excellent reference [50]. We may restrict all the variables of the problem to be natural numbers without loss of generality, see [50, p.6].

Finally, we give the definitions of Post's correspondence problem and its variations, and probabilistic finite automaton. Since they are used in Chapter 5 related to the words and matrix, some concepts used to define them can be found in Section 2.1.2.

Problem 4. [PCP] *Given a finite set of letters Σ, a binary alphabet Δ, and a pair of homomorphisms $h, g : \Sigma^* \to \Delta^*$, the Post's correspondence problem (PCP) is to determine whether there exist a word $w \in \Sigma^+$ such that $h(w) = g(w)$.*

Post's correspondence problem is a crucial problem in theoretical computer science. It was first introduced and shown to be undecidable by Emil Post [57]. The result was later improved in [51] that the undecidability still holds when $|\Sigma| = 7$, and it is decidable when $|\Sigma| = 2$ [31]. A recent result also claimed that for $|\Sigma| = 5$, PCP is also undecidable [53]. If the result is correct, then the decidability status when $3 \leq |\Sigma| \leq 4$ are currently open problems. PCP is widely used in undecidable proofs of problems for matrix semigroups (and elsewhere) due to its nondeterministic property. We give the following example of an instance of PCP.

Example 1. *PCP can be described as a type of puzzle over dominoes (or tiles). Let an individual domino contain a pair of words, one on top, one on bottom, which looks like, for example*

$$\left[\frac{b}{ab}\right].$$

Now given a collection of dominos,

$$\left\{\left[\frac{bba}{b}\right], \left[\frac{ab}{b}\right], \left[\frac{ba}{aabbaa}\right]\right\},$$

the task of the puzzle is to use the above dominos to find a match, or in other words, to make a list of these dominos (repetitions permitted) such that the word on the top is equal to the word on the bottom. In this example, we can find a match like

$$\left[\frac{bba}{b}\right]\left[\frac{ab}{b}\right]\left[\frac{ba}{aabbaa}\right]\left[\frac{ab}{b}\right],$$

as both top and bottom containing the word "bbaabbaab".

Many variations of PCP have been studied and some of them are also undecidable. We will require the following variant undecidable problem for proving later results.

Problem 5. [MMPCP] *Given a finite set of letters Σ, a binary alphabet Δ, and a pair of homomorphisms $h, g : \Sigma^* \to \Delta^*$, the Mixed Modification PCP (MMPCP)*

asks to determine whether there exists a word $w = x_1 \dots x_k \in \Sigma^+, x_i \in \Sigma$ *such that*

$$h_1(x_1)h_2(x_2)\dots h_k(x_k) = g_1(x_1)g_2(x_2)\dots g_k(x_k),$$

where $h_i, g_i \in \{h, g\}$*, and there exists at least one* j *such that* $h_j \neq g_j$*.*

Theorem 4. *[27] - The Mixed Modification PCP is undecidable for* $|\Sigma| \geq 9$*.*

It will later be useful to slightly modify the definition of this problem. As with other variants of Post's correspondence problem, the proofs of undecidability of the MMPCP often have the property that potential solution words are of the form $w = s_1 x_2 x_3 \cdots x_{k-1} s_{|\Sigma|}$, where $x_2, \dots, x_{k-1} \in \Sigma - \{s_1, s_{|\Sigma|}\}$, i.e. potential solution words must start with letter s_1, end with letter $s_{|\Sigma|}$, and all other letters are not equal to s_1 or $s_{|\Sigma|}$. An instance of the (MM)PCP which has this property is called a *Claus instance* of the problem. In fact all known proofs of the undecidability of (MM)PCP seem to have this property [35]. Claus instances can be useful for decreasing the resources required for showing certain undecidability results, and we use this property later in Chapter 5.

Theorem 5. *[35] - The Mixed Modification PCP is undecidable for Claus instances, when* $|\Sigma| \geq 9$*.*

To define and understand probabilistic finite automata, we first need several concepts. A vector $y \in \mathbb{Q}^n$ is a *probability distribution* if its elements are nonnegative and sum to 1 (y has an L_1 norm of 1). Matrix M is called a *column stochastic matrix* if each column is a probability distribution, a *row stochastic matrix* if each row is a probability distribution and it is called a *doubly stochastic matrix* if it is both row and column stochastic. For any row stochastic matrix M, if y is a probability distribution, then so is $y^T M$, since M preserves the L_1 norm on vectors and is nonnegative. The product of two row/column/doubly stochastic matrices is also row/column/doubly stochastic (respectively) as is not difficult to verify.

Definition 11. [PFA] *A Probabilistic Finite Automaton (PFA, see [21, 55] for further details) over an alphabet* A *is a triplet* (u, φ, v)*, where* $u \in \mathbb{Q}^n$ *is the initial probability distribution,* $\varphi : A^* \to \mathbb{Q}^{n \times n}$ *is a monoid homomorphism whose range*

is the set of n-dimensional row stochastic matrices and $v \in \mathbb{Q}^n$ *is the* final state vector *whose ith coordinate is* 1*, if state i is final, and* 0 *otherwise.*[2]

For a given PFA denoted $R = (u, \varphi, v)$ *and a word* $w \in A^*$*, we can define a function* $f_R : A^* \to [0, 1]$*, where:*

$$f_R(w) = u^T \varphi(w) v \in [0, 1]\,; \quad w \in A^*.$$

This is the probability of R being in a final state after reading word $w \in A^*$*.*

2.2 Computational Problems

2.2.1 Reachability Type Problems for Hybrid Automata

In this section we will define reachability and mortality problems for Hybrid Automata. The definition of reachability problem for HA is straightforward and easy to understand. The mortality, however, can be interpreted in different ways. We first define the reachability problem.

Problem 6. [Reachability for HA] *Given an HA* $\mathcal{H}$*, an initial configuration* $c = (l_0, \mathbf{x_0})$ *and a final configuration* $c' = (l_f, \mathbf{x_f})$*, the* reachability problem *is to determine if there exists a time* $0 < t < \infty$ *such that* $\lambda_{l_0,\mathbf{x_0}}(t) = l_f$ *and* $\xi_{l_0,\mathbf{x_0}}(t) = \mathbf{x_f}$*.*

Now we define the mortality problem for HA. Note that when we consider the mortality problem, we care about whether starting from any valid initial configurations, all the trajectories are "finite". By the word "finite" here we can define the trajectories halting within finite time or finite transitions. Also for all the finite trajectories, we can ask whether all the trajectories halt at the same configuration. Hence there is more than one possible way to define the mortality problem for HA. Let us consider four different ways below:

(1) All immortal trajectories are the ones that keep moving (the trajectories keep changing) with respect to time; all the others are the mortal trajectories which

[2]The definition of a PFA in the literature often interchanges the roles of u and v from our definition and requires column stochastic matrices, but the two can easily be seen to be equivalent by transposing all matrices and interchanging u and v.

halt in finite time.

(2) All the mortal trajectories are the ones that halt at one pre-defined configuration (the mortal configuration) in finite time; all the others are the immortal trajectories, which include the ones halting at some other configurations and the infinite ones with respect to time.

(3) All immortal trajectories are the ones that keep moving and have infinite transitions; all the others are the mortal trajectories which halt within a finite number of transitions.

(4) All the mortal trajectories are the ones that halt at one pre-defined mortal configuration within a finite number of transitions; all the others are the immortal trajectories, which include the ones halting at some other configurations and the ones having infinite transitions.

All these four ways make some sense. In (1) and (2) we differ finiteness and infiniteness by time while (3) and (4) we do this by the number of transitions. In (2) and (4) we define one particular mortal configuration while in (1) and (3) we do not.

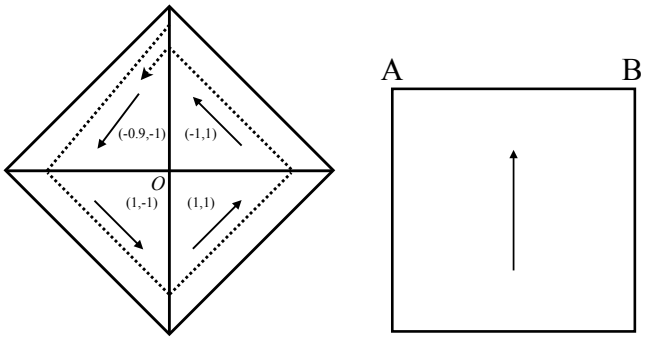


Figure 2.2: Two examples of the definition of mortality

See the example in Figure 2.2(a). In this system trajectories starting from any points will converge to the point $O = (0, 0)$ in finite time but with infinite transitions. So if we let O be the mortal point, in (1) and (2) it is a mortal system, but in (3) and (4) it is not. Then see the example in Figure 2.2(b), all

the trajectories will halt at the boundary AB. In (1) and (3) it is a mortal system while in (2) and (4) it is immortal.

As the main model studied in this thesis is the HPCD system, which is similar to the PCD model, we believe it is reasonable to define the case in Figure 2.2(b) being called mortal. Also we decide to define mortality problem with respect to time instead of transitions. As for dynamical systems, the mortality problem can be regarded as a finite case of the stability problem [20] (see Section 4.3 for more details about stability). So in this thesis we will define the mortality problem for HA by the first way mentioned above:

Problem 7. [**Mortality for HA**] *An HA $\mathcal{H}$ is called* immortal *if there exists at least one initial configuration $c = (l_0, \mathbf{x_0})$ for which there is an infinite trajectory starting at c, and such that for any $0 < t < \infty$, there exist $t < t_1 < \infty$ such that $\xi_{l_0,\mathbf{x_0}}(t) \neq \xi_{l_0,\mathbf{x_0}}(t_1)$. Otherwise, $\mathcal{H}$ is called* mortal*, in which case we say all the trajectories halt. The* mortality problem *is to determine if a given HA is mortal.*[3]

However, the other ways to define mortality also make sense. For example, to define a particular mortal configuration is similar to the mortality problem for Minsky Machine, which is to decide whether starting from any valid initial configuration, the machine will eventually halt at the mortal configuration [20]. Also stability problem for dynamical systems is defined based on an equilibrium point which is similar to a mortal configuration (see Section 4.3). On the other hand, defining the problem with respect to transitions instead of time will actually lead the study to an area called *Zeno behaviour*, which is a unique phenomenon to hybrid systems where a trajectory takes an unbounded number of discrete transitions in finite time, see [40] for example.

2.2.2 Reachability Type Problems in Matrix Semigroups

In this section we define two reachability type problems in matrix semigroups, called the scalar ambiguity problem and the scalar freeness problem. We first define them in the general case, and then we show a more restricted version over a bounded language of matrices.

[3]We call the problem mortality instead of immortality as it is common to call similar problems mortality for matrix semigroup and other dynamic systems [15, 19, 20, 24].

Consider a finite set $\mathcal{G} = \{G_1, G_2, \ldots, G_k\} \subset \mathbb{F}^{n\times n}$, generating a semigroup of matrices $\mathcal{S} = \langle \mathcal{G} \rangle$ and two column vectors $\rho, \tau \in \mathbb{F}^n$. Let $\Lambda(\mathcal{G})$ be the set of scalars such that $\Lambda(\mathcal{G}) = \{\lambda : \lambda = \rho^T M \tau | M \in \mathcal{S}\}$. If for $\lambda \in \Lambda(\mathcal{G})$ there exists a unique matrix $M \in \mathcal{S}$ such that $\lambda = \rho^T M \tau$, then we say that λ is *unambiguous* with respect to $\mathcal{G}, \rho, \tau$. $\Lambda(\mathcal{G})$ is called unambiguous if every $\lambda \in \Lambda(\mathcal{G})$ is unambiguous. If for $\lambda \in \Lambda(\mathcal{G})$ there exists a unique product $G_{i_1} G_{i_2} \cdots G_{i_m} \in \mathcal{S}$, with each $G_{i_l} \in \mathcal{G}$ such that $\lambda = \rho^T G_{i_1} G_{i_2} \cdots G_{i_m} \tau$, then we say that λ is *free* with respect to $\mathcal{G}, \rho, \tau$. $\Lambda(\mathcal{G})$ is called free if every $\lambda \in \Lambda(\mathcal{G})$ is free.

Problem 8. **[Scalar Ambiguity]** *Is $\Lambda(\mathcal{G})$ unambiguous with respect to $\mathcal{G}, \rho, \tau$?*

Problem 9. **[Scalar Freeness]** *Is $\Lambda(\mathcal{G})$ free with respect to $\mathcal{G}, \rho, \tau$?*

Problem 8 and Problem 9 look similar at first glance. However, the scalar ambiguity problem concentrates more on the properties of the semigroup $\mathcal{S}$ while the scalar freeness problem cares more about the properties of the set $\mathcal{G}$. A fact one can see from the definitions is that if the identity matrix I is contained in set $\mathcal{G}$, then the corresponding scalar set $\Lambda(\mathcal{G})$ is not free, but the same property does not hold for the scalar ambiguity problem. Also, we define the scalar freeness problem in a similar way of the matrix semigroup freeness problem. See the following two examples for further discussion.

Example 2. *Given a semigroup of matrices $\mathcal{S} = \langle \mathcal{G} \rangle$ generated by a finite set $\mathcal{G} = \left\{ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \right\}$ and two vectors $\rho = \tau = (1, 0)^T$, it is well-known that $\mathcal{S}$ is a free semigroup [27]. However, since*

$$1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

then scalar 1 is ambiguous with respect to $\mathcal{G}, \rho, \tau$ and thus $\Lambda(\mathcal{G})$ is ambiguous and not free even though $\mathcal{G}$ is free.

Example 3. *Given a semigroup of matrices $\mathcal{S} = \langle \mathcal{G} \rangle$ generated by a finite set $\mathcal{G} = \left\{ \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \right\}$, and two vectors $\rho = \tau = (1, 0)^T$, it is not difficult to*

verify that for $k \in \mathbb{N}$:

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}^k = \begin{pmatrix} 2^k & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}^k = \begin{pmatrix} 3^k & 0 \\ 0 & 1 \end{pmatrix}.$$

As the vectors ρ *and* τ *will only calculate the element* $M_{[1,1]}$ *for the matrix* $M \in \langle \mathcal{G} \rangle$*, every scalar in the set* $\Lambda(\mathcal{G})$ *is of the form* $2^m 3^n$*, where* $m, n \in \mathbb{N}$ *and* $m + n \neq 0$*. The only way to generate such a scalar by a single matrix is*

$$2^m 3^n = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 2^m 3^n & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

thus $\Lambda(\mathcal{G})$ *is unambiguous. However, since the two matrices in the set* $\mathcal{G}$ *are commutative, the semigroup* $\mathcal{S}$ *is clearly not free, and*

$$2^m 3^n = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}^m \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}^n \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}^n \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}^m \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

which indicates that $\Lambda(\mathcal{G})$ *is also not free. Notice that if we select a different pair of vectors, for example* $\rho = (1, 1)^T, \tau = (0, 1)^T$*, the scalar set* $\Lambda(\mathcal{G})$ *can become neither free nor unambiguious.*

Example 2 shows that a scalar set $\Lambda(\mathcal{G})$ can be ambiguous and not free even if $\mathcal{S} = \langle \mathcal{G} \rangle$ is a free semigroup. Example 3 shows that even if a scalar set $\Lambda(\mathcal{G})$ itself and the corresponding matrix semigroup $\mathcal{G}$ are not free, the scalar set can be unambiguous or not, depending on the vectors given. The links between the scalar ambiguity problem, scalar freeness problem and matrix semigroup freeness problem are illustrated by Proposition 3 in Section 5.1, Chapter 5.

We now restrict the the above two problems over a bounded language of matrices. Given a finite set of matrices $\{M_1, \ldots, M_k\} \subseteq \mathbb{Q}^{n \times n}$, we define *a bounded language of matrices* to be of the form:

$$\{M_1^{j_1} \cdots M_k^{j_k} | j_i \geq 0 \text{ where } 1 \leq i \leq k\}.$$

So we can define the restricted versions of these two problems as:

Problem 10. [**Scalar Ambiguity over a Bounded Language**] *Given k matrices $M_1, M_2, \ldots, M_k \in \mathbb{Q}^{n\times n}$, generating bounded language $M = M_1^* M_2^* \cdots M_k^*$, and two vectors $\rho, \tau \in \mathbb{Z}^n$, whether there exist l_1,l_2,...,l_k,r_1,r_2,...,$r_k \in \mathbb{N}$ such that*

$$\rho^T M_1^{l_1} M_2^{l_2} \ldots M_k^{l_k} \tau = \rho^T M_1^{r_1} M_2^{r_2} \ldots M_k^{r_k} \tau,$$

where $M_1^{l_1} M_2^{l_2} \ldots M_k^{l_k} \neq M_1^{r_1} M_2^{r_2} \ldots M_k^{r_k}$.

Problem 11. [**Scalar Freeness over a Bounded Language**] *Given k matrices $M_1, M_2, \ldots, M_k \in \mathbb{Q}^{n\times n}$, generating bounded language $M = M_1^* M_2^* \cdots M_k^*$, and two vectors $\rho, \tau \in \mathbb{Z}^n$, whether there exist l_1,l_2,...,l_k,r_1,r_2,...,$r_k \in \mathbb{N}$ such that*

$$\rho^T M_1^{l_1} M_2^{l_2} \ldots M_k^{l_k} \tau = \rho^T M_1^{r_1} M_2^{r_2} \ldots M_k^{r_k} \tau,$$

where $l_j \neq r_j$ for at least one j.

2.2.3 Summary of Problems

We list the problems studied in the thesis below:

- Reachability problem for
  - bounded 2-HPCDs;
  - nondeterministic 2-RHPCDs;
  - 3-RHPCDs;
  - unbounded 3-RHPCDs;
  - n-RHPCDs;
  - 1-PAM;
  - 1-PRM;
- Mortality problem for
  - 2-RHPCDs;
  - 3-RHPCDs;
  - unbounded 3-RHPCDs;

- Lyapunov and asymptotic stability problem for 4-HPCD;
- Scalar ambiguity problem for matrix semigroups
  - general case and over bounded languages;
- Scalar freeness problem for matrix semigroups
  - general case and over bounded languages.

Chapter 3

Reachability Problems for HPCDs

In this chapter we shall explore the reachability problem for variant HPCD systems (mostly in low dimensions). As stated in Section 1.1.1, our purpose is to find out the most powerful HPCD systems for which the reachability is decidable and the least powerful ones for which the reachability is undecidable.

We start with the 2-dimensional case, as that is how the HPCD was originally defined in [5]. We summarise the computational powers of 2-HPCD (see Section 2.1.3 for details) and show some complement results to the ones in [5].

We then study the reachability problem for a restricted version of HPCD systems. As the new model is highly restricted, we extend it to the n-dimensional case. Even so, we are still able to decidability for the reachability problem, and thus some complexity results are also given.

We show that a proof technique used in the complexity result for 3-RHPCDs can also be applied to show a complexity result for reachability for 1-PAM. Though the result itself is not new, the method might be interesting and we write it down for the interested reader. A more general class of 1-dimensional piecewise maps called 1-PRM is also discussed.

Finally we add unboundedness and nondeterminism to RHPCD systems, respectively, and different results are shown.

3.1 Restrictions of 2-HPCDs

In this section, we add some restrictions to the model of 2-HPCDs and explore the decidability of reachability problems for them. Our starting point is the model of 2-dimensional Restricted HPCD (2-RHPCD, see Section 2.1.3 for definitions). We first prove that a 2-RHPCD endowed with arbitrary constant flows can simulate a 1-PAM.

Before we stating a technical lemma, we introduce a terminology that will be used in this section:

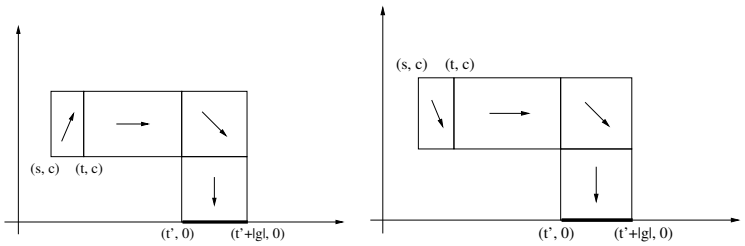
Mappings - A well-known technique for the analysis of PCDs is to study the edge-to-edge successor function [4], also called the Poincaré map [41] of the system. We will use a related concept in this section for HPCDs. Given an HPCD $\mathcal{H}$ and two line segments $L = [\mathbf{p_1}, \mathbf{p_2}]$ and $L' = [\mathbf{p'_1}, \mathbf{p'_2}]$, where $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p'_1}, \mathbf{p'_2} \in \mathbb{R}^{\mathbf{2}}$. We say that $\mathcal{H}$ *maps* L to L' in location l if for any $0 \leq \alpha \leq 1$, there exists a $t \geq 0$ such that for the trajectory defined over $[0, t]$, $\xi_{l,(\mathbf{p_1}+(\mathbf{p_2}-\mathbf{p_1})\alpha)}(t) = (\mathbf{p'_1} + (\mathbf{p'_2} - \mathbf{p'_1})\alpha)$ and if the symbolic dynamics of the trajectory is the same for any such choice of $0 \leq \alpha \leq 1$. Note that $L' = [\mathbf{p'_1}, \mathbf{p'_2}] = [\mathbf{p'_2}, \mathbf{p'_1}]$ and so the definition of mapping holds if we can map L to one of these two representations. A similar definition holds for when L is an open or half-open interval, mutatis mutandis. We call L and L' intervals by abuse of notation (if there is no confusion with rational intervals).

Lemma 1. *Given a 1-dimensional interval $I = (s, t)$, an affine function $f(x) = ax + b$ and a value m, where $a, b, m, s, t \in \mathbb{Q}$ are constants. Then there exists a 2-RHPCD system with arbitrary constant flows which maps $I \times \{0\}$ to $\{f(I) + m\} \times \{0\}$.*

Proof. We prove this lemma by 3 steps.

Step 1 - Interval $I \times \{0\}$ can be mapped to interval $I \times \{c\}$, where $c \in \mathbb{Q}^+$, by a bounded 2-PCD with non-comparative guards using flow $(0, 1)$.

Step 2 - Suppose we have an affine function $f(x) = ax + b$, and the 1-dimensional rational interval $I = (s, t)$. For any constant t' where $t' \geq t > s$, define $g = f(t) - f(s)$ and $s' = t' + |g|$. Assume that $c > |g| + |b| > 0$. Then we show the interval $I \times \{c\}$ can be mapped to $I' \times \{0\} = (t', s') \times \{0\}$ by a bounded 2-PCD system with non-comparative guards, see Figure 3.1. We need to consider 2 cases,



(a) $a > 0$ (b) $a < 0$

Figure 3.1: Lemma 1 Step 2: map $(s,t) \times \{c\}$ to $(t',s') \times \{0\}$.

$a > 0$ and $a < 0$. Note the 'orientation' of the interval will be reversed after the mapping.

1. $a > 0$. See Figure 3.1(a). We use flows $(1,a),(1,0),(1,-1)$ and $(0,-1)$ to map interval $(s,t) \times \{c\}$ to $(t',t'+|g|) \times \{0\}$.

2. $a < 0$. See Figure 3.1(b). We use flows $(1,a),(1,0),(1,-1)$ and $(0,-1)$ to map $(s,t) \times \{c\}$ to $(t',t'+|g|) \times \{0\}$. As we assume $c > |g| + |b| > 0$, so $c - |g| > |b| > 0$, which means the rectangle $\{(x,y) | t < x < t', c - |g| < y < |g|\}$ does not intersect with the x-axis.

Step 3 - Using a similar idea we can show the interval $I' \times \{0\} = (t',s') \times \{0\}$ can be mapped to $\{f(I) + m\} \times \{0\}$, where $\{f(I)\} = (f(s), f(t))$ if $a > 0$ and $\{f(I)\} = (f(t), f(s))$ if $a < 0$, by a bounded PCD system with non-comparative guards. We can use only the upper or lower half plane of the 2-PCD. Here we only prove the case when $a > 0$ and $f(t) + m < t'$ by using the lower half plane, other cases can be proven similarly.

(i) Use flow $(-1,-1)$ to map $(t',s') \times \{0\}$ to $\{\frac{1}{2}(t' + f(t) + m)\} \times (-\frac{1}{2}|t' - f(t) - m| - |g|, -\frac{1}{2}|t' - f(t) - m|)$;

(ii) Use flow $(-1,1)$ to map $\{\frac{1}{2}(t' + f(t) + m)\} \times (-\frac{1}{2}|t' - f(t) - m| - |g|, -\frac{1}{2}|t' - f(t) - m|)$ to $(f(s) + m, f(t) + m) \times \{0\}$.

Combining Steps 1, 2 and 3 we get the result of the lemma using a 2-location 2-RHPCD with arbitrary constant flows and non-comparative guards. In location 1 we realize Step 1 and jump to location 2, i.e., the guards are $s_i \leq x < t_i \wedge y = c$. In location 2 we realize Step 2 and Step 3 together because Step 2 only uses the upper plane of a 2-PCD and Step 3 only requires the lower plane of a 2-PCD. A similar proof holds for when I is an open or half-open interval, mutatis mutandis. □

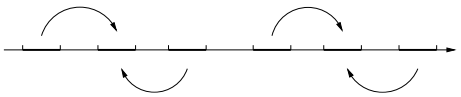


Figure 3.2: Idea of Theorem 6: map every two adjacent intervals into one interval

Theorem 6. *A 1-PAM with n intervals can be simulated by a 2-RHPCD with $\lceil \log_2 n \rceil + 3$ locations such that one of the variables has arbitrary constant flows.*

Proof. Suppose 1-PAM $\mathcal{A}$ is defined by $f(x) = a_i x + b_i$ if $x \in I_i$, with $1 \leq i \leq n$ and I_i are rational intervals. In the sequel, we assume all the intervals I_i in $\mathcal{A}$ are left closed and right open. Other cases can be proved similarly. Let the left and right endpoints of I_i be s_i and t_i respectively. First, we show that this 1-PAM can be simulated straightforwardly by an $(n+1)$-location 2-RHPCD with arbitrary constant flows. We need a single location p as the global state and n locations q_i for each interval I_i, $1 \leq i \leq n$.

1. In location p, we define the corresponding points of the 1-PAM $\mathcal{A}$ on interval $[s_1, t_n) \times \{0\}$. We then map each $I_i \times \{0\}$ to the interval $I_i \times \{c\}$, where $c = |\max\{|a_i|\}(t_n - s_1)| + \max\{|b_i|\}$. (See Lemma 1, Step 1). The transition guards of p are: $s_i \leq x < t_i \wedge y = c$, in which we jump to q_i.

2. In location q_i, map $I_i \times \{c\}$ to $\{f(I_i)\} \times \{0\}$ (see Lemma 1, Step 2&3). The transition guard of q_i is: $s_1 \leq x < t_n \wedge y = 0$, with a jump to location p.

The above method requires $n+1$ locations for a 1-PAM with n intervals. We now give an improved method using a 2-RHPCD with only $\lceil \log_2 n \rceil + 3$ locations.

Suppose the 1-PAM $\mathcal{A}$ contains n intervals. For every $n \neq 2^d$, $d \in \mathbb{N}$, there exists a minimum integer $k \in \mathbb{N}$ such that $\log_2(n+k) = \lceil \log_2 n \rceil$. The 1-PAM $\mathcal{A}$ can be expanded to $\mathcal{A}'$ such that $f(x) = a_i x + b_i$ if $x \in I_i$, where $i \in \{1, \ldots, n\}$. For every $i \in \{n+1, \ldots, n+k\}$, the length of each new added interval is given by $|I_{\varepsilon_i}| = \epsilon$, and the corresponding affine function is $f(x) = x$. This expansion does not change the dynamics of the 1-PAM $\mathcal{A}$, thus we assume $n = 2^d$, $d \in \mathbb{Z}$.

Again, let the left endpoint and the right endpoint of I_i be s_i and t_i respectively. Define c to be $c = |\max\{|a_i|\}(t_n - s_1)| + \max\{|b_i|\}$ and l to be $l = |t_n - s_1|$.

Step 1 Define the 1-PAM on interval $[s_1, t_n) \times \{0\}$. For every $i \in \{1, 2, ..., n\}$, map $I_i \times \{0\}$ to interval $I_i \times \{2(n-i+1)c\}$. (See Lemma 1, Step 1). In

this step each interval is mapped to a different height $y = 2(n - i + 1)c$. There is a $2c$-length 'gap' between every two intervals I_i and I_{i+1} and I_i is 'higher' than I_{i+1}. In Lemma 1 Step 2 this clearly prevents intersections in the following step.

Step 2 Map each interval $I_i \times \{2(n - i + 1)c\}$ to $\{f(I_i) + 2(n - i + 1)l\} \times \{0\}$. (See Lemma 1, Step 2). Then between every two intervals there is a 'gap' whose length is l.

Step 3 For i from 1 to $\frac{n}{2}$, let $j = 2i - 1$, we can find an undefined interval between $\{f(I_j) + 2(n - j + 1)l\} \times \{0\}$ and $\{f(I_{j+1}) + 2(n - j + 2)l\} \times \{0\}$ of length l. By the proof of Lemma 1 (Step 3), we can map $\{f(I_j) + 2(n - j + 1)l\} \times \{0\}$ using the upper plane and $\{f(I_{j+1}) + 2(n - j + 2)l\} \times \{0\}$ using the lower plane to this interval.

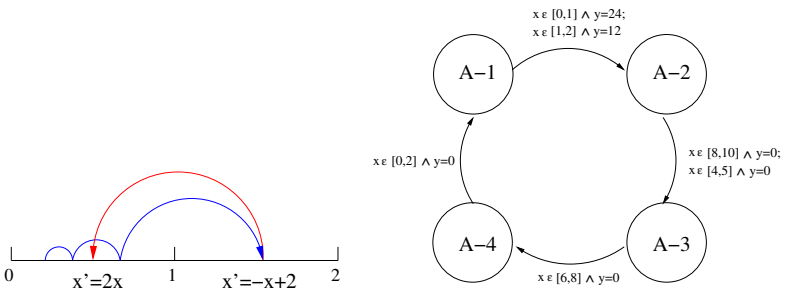
Step 4 Repeat Step 2 for $\log_2(n)$ times until only 1 interval, I_f, remains.

Step 5 If the orientation of I_f is 'reversed' with respect to the initial interval of the 1-PAM $\mathcal{A}$, then map I_f to this initial interval; otherwise, we reverse it before mapping it to the initial interval.

Step 1, 2 and 5 each requires 1 location. Step 3 and Step 4 require $\log_2 n$ locations, thus $(\log_2 n) + 3$ locations are required.

In this method every point w in the 1-PAM $\mathcal{A}$ is encoded by a point $(w, 0)$ in the interval $[s_1, t_n) \times \{0\}$ in the location of Step 1, including the initial and final points, and a one-step computation of $\mathcal{A}$ from point w to $f(w) = w'$ is represented by a trajectory segment of the 2-RHPCD from point $(w, 0)$ to $(w', 0)$ in the location of Step 1, calculated from Step 1 to Step 5. Thus it is a simulation and the statement of the theorem holds. □

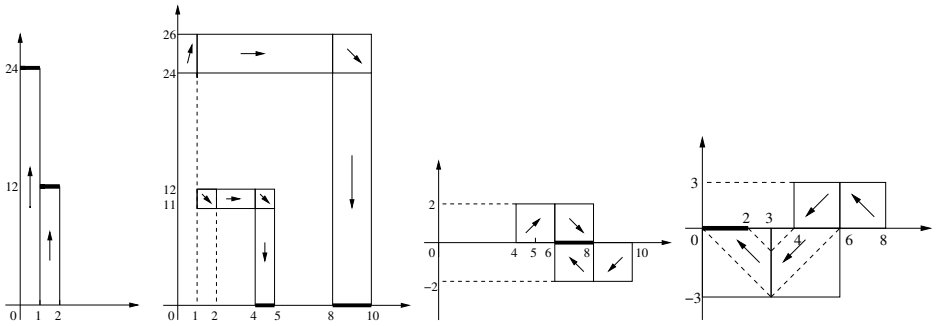
The difficulty of simulating a 1-PAM by a 2-PCD is that regions cannot overlap in a 2-PCD, i.e., one region has only one deterministic constant flow. Thus it is impossible to map several different intervals into a single interval under a 2-PCD, which suggests that to improve the lower bound $\Omega(\log_2 n)$ of the number of locations required to simulate an n-interval 1-PAM by a 2-RHPCD with arbitrary constant flows might be difficult.



(a) A 1-dim. Piecewise Affine Map

(b) 2-RHPCD with arbitrary constant flows

Figure 3.3: The 1-PAM with its equivalent 2-HPCD



(a) A-1 (b) A-2 (c) A-3 (d) A-4

Figure 3.4: The 2-PCDs of the 2-HPCD in Figure 3.3b (transition guards in bold).

Example 4. *We give an example of a 1-PAM below and show how to simulate it by a 2-RHPCD with arbitrary constant flows in Figures 3.3, 3.4.*

$$f(x) = \begin{cases} 2x, & if \quad x \in [0,1) \\ -x+2, & if \quad x \in [1,2] \end{cases}$$

Let the initial point be x_0. The initial location of the 2-HPCD is A-1, with variables $(x,y) = (x_0, 0)$. 2-PCD A-1 corresponds to Theorem 6, Step 1. 2-PCD A-2 separates each interval onto the x axis (Theorem 6, Step 2). 2-PCD A-3 combines together these two intervals (Theorem 6, Step 3). Finally, in A-4, as the final interval $[6,8]$ has the same orientation as the initial interval $[0,2]$, we reverse it before mapping it back to the initial interval (Theorem 6, Step 5).

We now show that a 2-RHPCD with linear resets can simulate a 1-PAM.

Lemma 2. *The interval $I \times \{0\}$ can be mapped to $\{f(I)+m\} \times \{0\}$ by a 2-RHPCD*

system with linear resets, where $f(x) = ax + b$ is an affine function, $I = (s, t)$ is a 1-dimensional interval and $a, b, m, s, t \in \mathbb{Q}$ are constants.

Proof. The proof is similar to the proof of Lemma 1.

Step 1 First map the interval $I \times \{0\}$ to the interval $I \times \{c\}$ by flow $(0, 1)$. Define the transition guard to be $I \times \{c\}$, which jumps to location 2 with linear reset: $x' = |a|x, y' = y$.

Step 2 Using the similar idea in Lemma 1 Step 2, we can map the interval $|a|I \times \{c\}$ to the interval $(t', t' + |g|) \times \{0\}$ by the flows $(1, 1)$ if $(a > 0)$ or $(1, -1)$ if $a < 0$, $(1, 0)$, $(1, -1)$ and $(0, -1)$, where t' and g are defined the same as in Lemma 1.

Step 3 Exactly the same as Lemma 1 Step 3. □

Theorem 7. *A 1-PAM with n intervals can be simulated by a 2-RHPCD containing $\lceil \log_2 n \rceil + 3$ locations with linear resets.*

Proof. Apply Lemma 2 instead of Lemma 1 in the proof of Theorem 6. □

Corollary 1. *A 2-RHPCD with linear resets and a 2-RHPCD with affine resets can simulate each other.*

Proof. Immediately from the results of [5] and Theorem 7. □

Definition 12. [**1-POM**] *Let f be a 1-PAM. We call f a* 1-dimensional piecewise offset map *(1-POM) if $f(x) = x + b_i$ for all $x \in I_i$.*

Corollary 2. *A 1-POM can be simulated by a 2-RHPCD, and a 2-RHPCD can be simulated by a 1-POM.*

Proof. The first part follows immediately from Theorem 6. As any coefficient of the linear part of a 1-POM is 1, only elementary flows are required for simulating a 1-POM by a 2-RHPCD. The second part is from [5]. □

The following theorem shows a relationship between the additional computational powers of affine resets and arbitrary constant flows.

Theorem 8. *A k-location 2-RHPCD with arbitrary constant flows can be simulated by a k-location 2-RHPCD with affine resets for any $k \geq 1$.*

Proof. Immediately from Lemma 1, Lemma 2 and Corollary 1. □

3.2 Higher dimensional RHPCDs

In this section, we start by showing that reachability is co-NP-hard for 3-RHPCDs by an encoding of the simultaneous incongruences problem (see Problem 2). Although this bound may seem quite limited, recall that the system is deterministic, which substantially limits its power. We also show how the proof technique can be used in a complexity result for 1-PAM, and an undecidable result for a related model 1-PRM is given. We later show that reachability is in PSPACE for bounded n-RHPCDs, for any $n \geq 1$. We start with a technical lemma.

Lemma 3. *There exist solutions for the simultaneous incongruences problem with a collection* $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ *if and only if there exists a solution* k *such that* $0 < k \leq \rho$, *where* $\rho = lcm(b_1, \ldots, b_n)$ *and* $lcm(b_1, \ldots, b_n)$ *is the least common multiple of* $b_1, \ldots, b_n$.

Proof. The sufficient part is trivial. We show the necessary part. Given an instance $\{(a_1, b_1), \ldots, (a_n, b_n)\}$, let $\rho = \text{lcm}(b_1, \ldots, b_n)$. Then for every $1 \leq i \leq n$, $\rho \equiv 0 \pmod{b_i}$.

For every integer $k > \rho$, we can rewrite k as $k = k_0 + m\rho$, where $0 < k_0 \leq \rho$ and $m \in \mathbb{N}$. Suppose there exists a solution $k_s > \rho$. According to the simultaneous incongruences problem, we know that $k_s \not\equiv a_i \pmod{b_i}$ for all i, where $1 \leq i \leq n$. So we can find a k_0, where $0 < k_0 \leq \rho$, and a positive integer m such that

$$k_s \equiv k_0 + m\rho \not\equiv a_i \pmod{b_i},$$

for every i, where $1 \leq i \leq n$. But $\rho \equiv 0 \pmod{b_i}$ for all $1 \leq i \leq n$, thus

$$k_0 \not\equiv a_i \pmod{b_i}$$

for all $1 \leq i \leq n$, thus k_0 is the solution we want. □

Theorem 9. *The reachability problem for bounded* 3*-RHPCDs is co-NP-hard.*

Proof. Consider an instance of the simultaneous incongruences problem with n pairs. We will encode the instance into a reachability problem for a 3-RHPCD. Starting from $k = 1$, we test whether $k \bmod b_i \neq a_i$ holds for each pair (a_i, b_i). If

it does hold for every i, then the current value of k is the solution. If for some i we find $k \bmod b_i = a_i$, then the current value of k is not a potential solution. We increase the value of k by 1 and start the testing all over again. By Lemma 3 there are at most ρ integers to test.

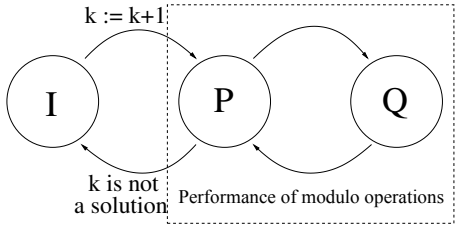
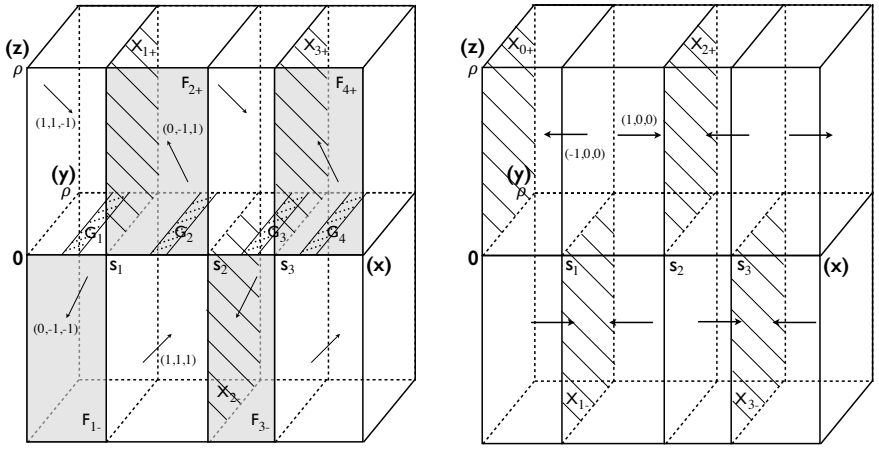


Figure 3.5: Reachability for 3-RHPCD (location I actually represents 3 locations I_1, I_2 and I_3)



(a) Location P (b) Location Q

Figure 3.6: 3-RHPCD encoding simultaneous incongruences problem (only location P and location Q are shown)

We construct the corresponding 3-RHPCD in the following way. We define 5 locations P, Q, I_1, I_2 and I_3. Locations P and Q together can 'perform' the modulo operation test for a certain value of k and every pair of (a_i, b_i). Locations I_1, I_2 and I_3 can increase the value of k by 1 when we find the current k is not a potential solution. See Figure 3.5. Define regions A_i and B_i in locations P and Q :

$$A_i = (s_{i-1}, s_i) \times (0, \rho) \times (0, \rho);$$
$$B_i = (s_{i-1}, s_i) \times (0, \rho) \times (-\rho, 0),$$

where $i \in \{1, 2, ..., n\}$, $s_0 = 0$, $s_i = \sum_1^i b_i$ for $1 \leq i \leq n$, and $\rho = \text{lcm}(b_1, \ldots, b_n)$.

We call a region *odd* (resp. *even*) A_i or B_i if i is odd (resp. *even*). We also define surface O :

$$O = [0, s_n] \times [0, \rho] \times \{0\}.$$

To carry out the modulo operation for a certain pair (a_i, b_i), we use the regions odd $\overline{A}_i$ and even $\overline{B}_i$ in both locations P and Q. Define the derivative to be $(1, 1, -1)$ in odd $\overline{A}_i$ in P ($(1, 1, 1)$ in even $\overline{B}_i$ in P) and $(-1, 0, 0)$ in both odd $\overline{A}_i$ and even $\overline{B}_i$ in Q. See Figure 3.6. Intuitively, we arrange the regions alternately above and below the O surface instead of stacking them together. This is to avoid them sharing a common surface, which may cause nondeterminism when we define a (transition) guard on that surface.

For a point (x, y, z), we use the z coordinate to represent the current value of k and the y coordinate as a memory. Assuming i is odd (see Table 3.1 for full details of both odd and even cases), we start at point $\mathbf{x}_0 = (s_{i-1}, 0, k)$ in P and move according to the flow $\dot{\mathbf{x}} = (1, 1, -1)$. While $|z| > 0$, every time when $x = b_i + s_{i-1} = s_i$, we jump to Q. In Q we keep variables y and z unchanged, simply reset x to 0 by the flow $\dot{\mathbf{x}} = (-1, 0, 0)$ and jump back to P. Each time the trajectory goes from P to Q and jumps back to P, the absolute value of variable z will be subtracted by b_i. So when the trajectory hits the O surface (i.e., $z = 0$), the value of x will be equal to $s_{i-1} + (k \bmod b_i)$. Since y and z in P change at the same rate, when the absolute value of z drops from k to 0, the value of y will increase from 0 to k.

If $k \bmod b_i \neq a_i$, we reset y to 0 and $|z|$ to k by switching the value of these two variables, and enter region $\overline{B}_{(i+1)}$ to test whether $k \bmod b_{i+1} \neq a_{i+1}$. To do this, we use the regions odd B_i and even A_i in both locations P and Q. Define the derivative to be $(0, -1, -1)$ in odd B_i in P ($(0, -1, 1)$ in even A_i in P) and $(1, 0, 0)$ in both odd B_i and even A_i in Q. By the flows in P the value of y and $|z|$ are switched. When $y = 0$ we jump to Q and reset x to s_i, and then jump back to P to start testing the case of pair (a_{i+1}, b_{i+1}).

If $k \bmod b_i = a_i$, which means that the current value of k is not a potential solution, we jump to locations I_1, and then I_2 and I_3, (defined in Table 3.1) which moves the trajectory to point $(0, 0, k + 1)$ and 'restarts' in location P to test

whether the new value $k+1$ is a correct solution [1]. A correct solution k should satisfy that the trajectory starts from point $(0,0,k)$ in location P and can finally reach some point (in location P) on the surface $(s_{n-1}, s_n) \times (0, \rho) \times \{0\}$ with $x \notin (s_{n-1} + a_n - \frac{\varepsilon}{2}, s_{n-1} + a_n + \frac{\varepsilon}{2})$.

Location	Support Set	Flows	Guards
P	$\overline{A} \cup \overline{B}$	$\overline{A}_i$ (i is odd): $(1,1,-1)$ $A_i \cup F_{i+}$ (i is even): $(0,-1,1)$ $B_i \cup F_{i-}$ (i is odd): $(0,-1,-1)$ $\overline{B}_i$ (i is even): $(1,1,1)$	$\overline{X}_{i+}$ $(i = 1,3,...,n-1)$, $\overline{X}_{i-}$ $(i = 2,4,...n)$, F_{i+} $(i = 2,4,...,n)$, F_{i-} $(i = 1,3,...,n-1)$: jump to Q
			G_i : jump to I_1
Q	$\overline{A} \cup \overline{B}$	$\overline{A}_i$ (i is odd): $(-1,0,0)$ $A_i \cup F_{i+}$ (i is even): $(1,0,0)$ $B_i \cup F_{i-}$ (i is odd): $(1,0,0)$ $\overline{B}_i$ (i is even): $(-1,0,0)$	$\overline{X}_{i+}$ $(i = 0,2,...,n-2)$, $\overline{X}_{i-}$ $(i = 1,3,...,n-1)$: jump to P
I_1	$\overline{A}$	$(-1,0,0)$	$x = 0$ jump to I_2
I_2	$\overline{A}$	$(0,0,1)$	$z = 1$ jump to I_3
I_3	$\overline{A}$	$(0,-1,1)$	$y = 0$ jump to P

Table 3.1: Reachability problem for 3-RHPCD

We now give the formal details of this construction. Without loss of generality, we assume n is even. Define 2 regions A and B :

$$A = \cup_1^n A_i;$$
$$B = \cup_1^n B_i.$$

Also define four types of surfaces F_{i+}, F_{i-}, X_{i+} and X_{i-} :

$$\begin{aligned}
F_{i+} &= (s_{i-1}, s_i) \times \{0\} \times (0, \rho), && i = 1,2,...,n;\\
F_{i-} &= (s_{i-1}, s_i) \times \{0\} \times (-\rho, 0), && i = 1,2,...,n;\\
X_{i+} &= \{s_i\} \times (0, \rho) \times (0, \rho), && i = 0,1,2,...,n;\\
X_{i-} &= \{s_i\} \times (0, \rho) \times (-\rho, 0), && i = 0,1,2,...,n.
\end{aligned}$$

[1]Note that here in the guards we do not require exactly $x = a_i + s_{i-1}$, but allow some error ε, so tiny perturbations will not affect our result. The same analysis can be applied to Theorem 13. This implies that the system has robust reachability and mortality problems, but we do not expand on the details here. See more details about robustness in [39].

Finally, we define a set of ε-width strips G_i:

$$G_i = (s_{i-1} + a_i - \tfrac{\varepsilon}{2}, s_{i-1} + a_i + \tfrac{\varepsilon}{2}) \times [0, \rho] \times \{0\}, \qquad i = 1, 2, ..., n;$$

With the help of these notations, we construct the 3-RHPCD in Table 3.1.

The number of regions and guards in the constructed 3-RHPCD is clearly polynomial in the number of pairs of the simultaneous incongruences problem. Furthermore, the points defining each such region can be represented in binary and are therefore polynomial in the description size of the simultaneous incongruences problem. □

Proposition 1. *The reachability problem for bounded n-RHPCDs is in PSPACE.*

Proof. Given an n-RHPCD $\mathcal{H}$, an initial configuration $(q_0, \mathbf{x_0})$ and a final configuration $(q_f, \mathbf{x_f})$, we first show that starting from $(q_0, \mathbf{x_0})$, the trajectory will hit the internal and transition guards finitely many times before either reaching $(q_f, \mathbf{x_f})$, or detecting a cycle, or hitting some endpoints (at which time the calculation halts), thus 'convergence' to a point is not possible.

By the definition of n-RHPCD (see Definition 3, 4), the guards of $\mathcal{H}$ are of the form

$$\left(\bigwedge_{1 \le i \le n \wedge i \ne j} (a_i \prec x_i \prec' b_i) \right) \wedge (x_j = c_j)$$

where $j \in \{1, \ldots, n\}, x_i, x_j, a_i, b_i, c_j \in \mathbb{Q}$, and $\prec, \prec' \in \{<, \le\}$.

By definition, the components of $\mathbf{x_0} = (x_{0_1}, \ldots, x_{0_n})$ and $\mathbf{x_f} = (x_{f_1}, \ldots, x_{f_n})$ are rational numbers, i.e., $\mathbf{x_0}, \mathbf{x_f} \in \mathbb{Q}^n$. Define γ to be the least common multiple of all the denominators of the constants appearing in the description the n-RHPCD $\mathcal{H}$ (i.e. the guards, invariants, initial and final points) and the continuous components of the initial and final configurations $\mathbf{x_0}, \mathbf{x_f}$. Multiply all such constants by $\gamma \in \mathbb{N}$, i.e., let

$$A_i = \gamma a_i, B_i = \gamma b_i, C_j = \gamma c_j, \mathbf{X_0} = \gamma \mathbf{x_0}, \mathbf{X_f} = \gamma \mathbf{x_f}.$$

Thus, $A_i, B_i, C_j \in \mathbb{Z}$ and $\mathbf{X_0}, \mathbf{X_f} \in \mathbb{Z}^n$. Define a new n-RHPCD $\mathcal{H}'$ with initial configuration $(q_0, \mathbf{X_0})$ and final configuration $(q_f, \mathbf{X_f})$ by replacing a_i, b_i, c_j, $\mathbf{x_0}, \mathbf{x_f}$ by $A_i, B_i, C_j, \mathbf{X_0}, \mathbf{X_f}$. Clearly, $\mathcal{H}$ reaches $\mathbf{x_f}$ iff $\mathcal{H}'$ reaches $\mathbf{X_f}$, and $\mathcal{H}'$ is described by integer values only.

For $\mathcal{H}'$, the trajectory starts at integer configuration $\mathbf{X_0}$, and the guards of $\mathcal{H}'$ are defined by integers. Since all the flows of $\mathcal{H}'$ are chosen from the set $\{0, 1, -1\}$, when one variable x_i of a point of the trajectory, $\mathbf{X_t}$, changes its value from one integer to another, any other variable x_j of $\mathbf{X_t}$ remains an integer. So every time the trajectory hits a guard, i.e., the condition $(\bigwedge_{1\leq i\leq n \wedge i\neq j}(A_i \prec x_i \prec' B_i)) \wedge (x_j = C_j)$ is satisfied by the components of $\mathbf{X_t}$, $\mathbf{X_t}$ will have integer components.

We now prove that the problem can be solved in PSPACE. Note that the representation size of γ is clearly polynomial in the representation size of $\mathcal{H}$, thus so is the size of $\mathcal{H}'$. We now show that the representation size of the number of possible transition configurations (the configuration when the trajectory hits the guard and takes transition) of $\mathcal{H}'$ is also polynomial in the size of $\mathcal{H}$.

Let $k > 0$ be the number of locations of $\mathcal{H}'$. Since $\mathcal{H}$ is bounded, we can calculate $\tau \in \mathbb{N}$ to be the maximal absolute value of the endpoint of any invariant of $\mathcal{H}$ over all locations. Thus the range of variables of $\mathcal{H}'$ is contained within $[-\gamma\tau, \gamma\tau]$. Since we have n variables, the maximal number of transition configurations of $\mathcal{H}'$, starting at initial configuration $(q_0, \mathbf{X_0})$ is thus $k(2\gamma\tau)^n$, which can be represented in size polynomial in the size of $\mathcal{H}$, since it requires at least $k\log((\gamma\tau)^n) = nk\log(\gamma\tau)$ space to store $\mathcal{H}$ and

$$\frac{\log(k(2\gamma\tau)^n)}{nk\log(\gamma\tau)} = \frac{\log(k) + n\log(2\gamma\tau)}{nk\log(\gamma\tau)} < c$$

for some computable constant $c > 0$. We can use a counter to keep track of the number of transitions the trajectory of $\mathcal{H}'$ makes, starting from $(q_0, \mathbf{X_0})$. As each transition is taken, we can determine if the final configuration was reached since the last transition. Otherwise, we increment the counter and proceed. If the counter reaches $k(2\gamma\tau)^n$, then the configurations must be periodic and we can halt. Hence the reachability problem is in PSPACE. □

3.3 1-PAM and 1-PRM

In this section we first show reachability problem for 2-HPCD is co-NP-hard by encoding of an instance of a simultaneous incongruences problem. Thus reachability for 1-PAM is also co-NP-hard, since the reduction from a 2-HPCD to a 1-PAM

can be done in polynomial time and space [5]. Though it is known in [19] that the reachability for 1-PAM over integers is PSPACE-hard, which is a stronger result, we still think our encoding method is interesting and worth to show.

Theorem 10. *The reachability problem for bounded* 2*-HPCD is co-NP-hard.*

Proof. Given an instance of the simultaneous incongruences problem with n pairs $\{(a_1, b_1), \ldots, (a_n, b_n)\}$, we encode the instance into a reachability problem for a 2-HPCD. For an initial integer k, we test whether $k \bmod b_i \not\equiv a_i$ holds for each pair (a_i, b_i) by the corresponding location P_i in the 2-HPCD. If the inequation does hold for every i for the current value k, then a trajectory of the 2-HPCD starting from P_1 is able to go through every location P_i and Q_i (which is used to reset the value of current k) and reach the final location Y, which means there exists a solution for the problem. If the incongruence does not hold for some i, then the current k is not a potential solution. We goes to location R, decrease the value of k by 1 and start the test all over again (see Fig. 3.7).

Let $\rho = \mathrm{lcm}(b_1, \ldots, b_n)$, by Lemma 3 starting from integer ρ there are at most ρ integers to test.

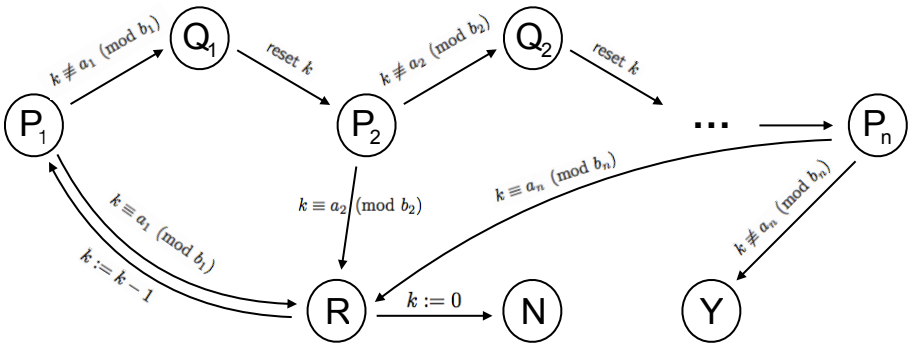


Figure 3.7: 2-HPCD simulating simultaneous incongruences problems

In the 2-HPCD we use the x-coordinate to store the current value of k and the y-coordinate as a timer. To avoid losing the value of k after the modulo operation in location P_i, we also encode k into the fractional part of the x-coordinate. Suppose ρ has m bits in its binary representation, we encode k as $(1 + 2^{-m})k$.

The 2-HPCD consists of $2n + 3$ locations, labelled R, N, Y and P_i, Q_i for $1 \leq i \leq n$. Intuitively, Y is the Yes location (solution found), N is the No location (no solution found). Location P_i is used to test whether for $x = (1 + 2^{-m})k$, we have

that $k \not\equiv a_i \mod b_i$. Q_i is used to reset $l + 2^{-m}k$ (where $l \leq b_i$) to $(1 + 2^{-m})k$ for the next test. Finally, R is the reset location to reset x to $(1 + 2^{-m})(k - 1)$ and start the procedure again, if k was not a solution.

In location P_i, the trajectory starts from point $((1+2^{-m})k, 0)$, with derivative $(0, 1)$. When $y = 1$, the guard is activated and if $x \in [b_i, \rho + 1)$, then we subtract b_i from the x-coordinate by the reset $x := x - b_i$ and $y := 0$. Eventually, we hit the guard $y = 1$ and $x \in [0, b_i)$, at which point we determine whether $k \bmod b_i \not\equiv a_i$ by testing whether $x \in [a_i, a_i + 1)$ or not.

If x is not in the interval $[a_i, a_i+1)$, it means the current value of k is a potential solution. This is because x was initially $(1 + 2^{-m})k$ and in P_i we have calculated that $(1 + 2^{-m})k \mod b_i \notin [a_i, a_i + 1)$, which implies that $k \mod b_i \not\equiv a_i$. We should thus test it for the next pair (a_{i+1}, b_{i+1}). Before starting the next test we reset k in location Q_i. In Q_i, we start at some point $(x, 0)$. When the guard $y = 1$ is activated, if $x \in [1, \max\{b_i\})$ then we subtract 1 from x. When x is in the interval $[0, 1)$, we reset the value of k by the function $x := (2^m + 1)x$ and jump to location P_{i+1}.

If x is in the interval $[a_i, a_i + 1)$, then the current value of k we are testing is not a potential solution. This is because we have calculated that $k \mod b_i \equiv a_i$. We thus jump to location R to reset k to $k - 1$ and start the next round of tests for the new value. Location R is similar defined as location Q_i except that the (affine) reset function is $x := (2^m + 1)x - (1 + 2^{-m})$, which then jumps to location P_1.

Clearly, all components of the 2-HPCD has a size polynomial in the instance size of the simultaneous incongruence problem, since they are based on either m or n.

We now describe the full details of the 2-HPCD in Table. 3.2 by the following

notations:

$$A_i := [a_i, a_i + 1) \times \{1\}$$
$$B_i := [b_i, \rho + 1] \times \{1\}$$
$$C_i := [0, b_i) \setminus [a_i, a_i + 1) \times \{1\}$$
$$D := (0, 1) \times \{1\}$$
$$E := [1, \max\{b_i\} \times \{1\}$$
$$U := [0, \rho + 1] \times [0, 1]$$
$$V := [0, \max\{b_i\}] \times [0, 1].$$

Location	Support Set	Flows	Guards & Resets
P_i	U	$(0, 1)$	$B_i : (x, y) := (x - 1, 0)$, jump to P_i C_i : jump to Q_i C_n : jump to Y (for P_n) A_i : jump to R
Q_j	V	$(0, 1)$	$E : (x, y) := (x - 1, 0)$, jump to Q_i $D : (x, y) := (10^m + 1)x, 0)$, jump to P_{i+1}
R	V	$(0, 1)$	$E : (x, y) := (x - 1, 0)$, jump to R $D : (x, y) := ((2^m + 1)x - (1 + 2^{-m}), 0)$, jump to P_1 $x = 0$: jump to N

Table 3.2: 2-HPCD encoding simultaneous incongruence problems

□

Corollary 3. *The reachability problem for* 1*-PAM is co-NP-hard.*

NOTE: It should be noticed that we could not adapt this method to make it work for a proof that shows co-NP-hardness of mortality for 1-PAM (whether all points eventually reach some fixed point, for example the origin). This is due to the following reason. To show mortality is co-NP-hard, we need to build a similar model as above with a few adaptions, such that a loop occurs during the run if and only if there exists a valid solution to the corresponding simultaneous incongruences problem, and all other 'wrong answers' will lead to a mortal location where the calculation halts. In the reachability proof, we use a fixed length fractional part to help us store the value of k we are testing. This is because in the reachability problem, we can choose the initial configuration to start with. However, in the mortality problem, the calculation could begin with any valid point, which means the length of the fractional part could be arbitrarily long. If the same method is

used again, it is not difficult to find a counterexample which does not encode the correct answer for the simultaneous incongruences problem but still cause a loop. Unfortunately, we don't know how to fix this problem.

Further discussion: It is well known that the decidability of reachability and mortality for 1-PAM is a longstanding open problem [5, 19, 20, 43, 45]. From the above discussion we know that even proving co-NP-hardness of mortality for 1-PAM is difficult. The example below highlights the difficulty.

The failure of the proof of the co-NP-hardness of mortality for 1-PAM comes from the arbitrary length and value of fractional part or the input rational number. One may wonder if this can be solved by showing that, for a 1-dimensional piecewise system, whether a periodic point can lead to a periodic dense interval. Or, formally speaking, whether the following assumption is true:

Problem 12. *Let A be a periodic point in a 1-dimensional piecewise system, does there exist an $\varepsilon > 0$, such that all points in $(A-\varepsilon, A]$ or $[A, A+\varepsilon)$ are also periodic?*

Unfortunately, we can find a counterexample to show that it is not true. See the example below:

$$h(x) = \begin{cases} rx, & \text{if} \quad x \in [0, \frac{1}{2}) \\ -r(x-1), & \text{if} \quad x \in [\frac{1}{2}, 1] \\ -1, & \text{otherwise} \end{cases}$$

where $r > 2$ can be any rational number. Let $h^{(n)}(x)$ denotes the function $h(x)$ being applied for n iterations, and s_1 and s_2 (assume $s_1 < s_2$) denote the two roots of the function $h(x) = 1$. It is easy to verify that $s_1, s_2 \in [0, 1]$. Then we can see that all the points in the intervals $I_1 = [0, s_1]$ and $I_2 = [s_2, 1]$ will be mapped back to $[0, 1]$ by $h(x)$, and all the other points in $[s_1, s_2]$ will be mapped to -1. See Fig. 3.8. Let

$$\Lambda = \{x | h^{(n)}(x) \in [0, 1]\},$$

where $n \in \mathbb{N}$. Then Λ represents the set of all points remaining in the interval $[0, 1]$ after n times of iterations. As $n \to \infty$, Λ forms a Cantor set. From the properties

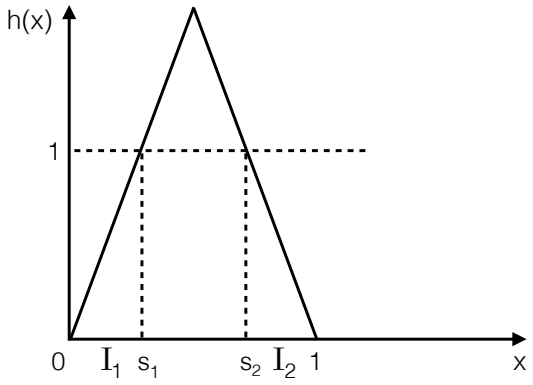


Figure 3.8: A system generating a Cantor set

of a Cantor set [9], we know:

(i) Λ is a closed, bounded set containing no intervals of nonzero length.

(ii) $h^{(n)}(x)$ has 2^n periodic points with period n.

(iii) The periodic points of $h^{(n)}(x)$ are dense in $[0, 1]$.

Clearly, $h(x)$ is a counterexample of the assumption above. It suggests that finding a decidability or a complexity result of mortality for the 1-dimensional piecewise systems is not easy. We may need an alternative method to eventually solve this problem.

1-Piecewise Rational Maps

It was shown in [45] that the reachability for 1-dimensional piecewise rational maps is undecidable. The proof is done by an encoding of a Minsky machine, which requires a 1-PRM of degree 2 (i.e. containing the term x^2) with a finite number of intervals defined on the entire real line $(-\infty, +\infty)$. Here we show the same result using a different method - by encoding the Generalised Collatz Problem. By this method we still require a 1-PRM of degree 2 but defined on the non-negative real line $(0, +\infty)$ only.

Note that here we use a slightly different definition from [19, 47], where the GCP is defined as whether *every* trajectory of g reaches 1. However, from the proof techniques used in [29, 47], we can see that Theorem 1 still holds.

Theorem 11. *The reachability for* 1*-PRM is undecidable.*

Proof. Given a GCP of standard representation $\{m, a_1, \ldots, a_m, b_1, \ldots, b_m\}$, the problem asks starting from an arbitrary positive integer k, whether the trajectory reaches 1. We will construct a 1-PRM with $m+2$ intervals such that starting from the point $k+\frac{1}{k}$, the trajectory of the 1-PRM reaches 1 if and only if the trajectory of the corresponding GCP reaches 1 starting from 1.

First we encode k into the point $k+\frac{1}{k}$ in the 1-PRM. To encode the modulo operation that is applied on k in the GCP, in the 1-PRM we define the function to be $f_{m+1}(x) = x-m$ on the interval $(m+1, +\infty)$. In other words, if k is greater than $m+1$, we keep subtract m from k until the new value is smaller than $m+1$, which simulate the modulo operation. The fractional part is used as a memory to store the value of k. After a finite number of steps, the trajectory will leave the interval $(m+1, +\infty)$, reaches a point $i+\frac{1}{k}$, $1 \leq i \leq m$, which falls into one of the intervals $(i, i+1)$. In the interval $(i, i+1)$, we need to simulate the operation $g_i(x) = a_i(x-i)/m + b_i$ of the generalise Collatz Function $g(x)$, as well as encode the new value of x into the fractional part. This can be done by:

$$f_i(x) = \frac{a_i((x-i)^{-1}-i)}{m} + b_i + (\frac{a_i((x-i)^{-1}-i)}{m} + b_i)^{-1}$$

The $x-i$ term maps $i+\frac{1}{k}$ to $\frac{1}{k}$, then we take the reverse of it and get the value of k, for which we apply the function $g_i(x)$. Clearly the last term encodes the new value into the fractional part. Under some computation, we can rewrite the function as:

$$\begin{aligned} f_i(x) &= \frac{cx+d}{m(x-1)} + \frac{m(x-i)}{cx+d} \\ &= \frac{(c^2+m^2)x^2 + 2(cd+me)x + d^2 + e^2}{cmx^2 + (dm+ce)x + de}, \end{aligned}$$

where $c = -i \cdot a_i, d = m \cdot b_i$ and $e = -i \cdot m \cdot b_i$. Then the corresponding 1-PRM $f(x)$ can be defined as:

$$f(x) = \begin{cases} x, & \text{if} \quad x \in (0,1]; \\ f_i(x), & \text{if} \quad x \in (i, i+1); \\ x-m, & \text{if} \quad x \in (m+1, +\infty); \end{cases}$$

where $1 \leq i \leq m, f_i(x)$ is defined as above. Clearly, if starting from the point $k + \frac{1}{k}$, the trajectory of the 1-PRM can reach the point 1, then the corresponding GCP has a solution. Thus the reachability for 1-PRM is undecidable. □

3.4 Extensions of RHPCDs

Theorem 12. *The reachability problem is undecidable for unbounded* 3*-RHPCDs.*

Proof. Consider a two counter (Minsky) machine $\mathcal{M}$, with set of instructions $\{p_i\}$ and two counters c_1 and c_2. For configuration (p_i, c_1, c_2), we define two locations P_i and T_i in an unbounded 3-RHPCD to encode instruction p_i. There are 3 'types' of instruction, where c_k represents a counter ($k \in \{1, 2\}$):

Type I - p_i: `Add` 1 `to` c_k; `goto` p_j;

Type II - p_i: `If` $c_k \neq 0$ `then subtract` 1 `from` c_k; `goto` p_{j_1};
`else goto` p_{j_2};

Type III - p_i : `Halt`.

Given a vector $\mathbf{x} = (x, y, z)$ in an unbounded 3-RHPCD, we use variable x to represent the counter c_1, y to represent the counter c_2 and z as a timer which ensures x or y increases or decreases by exactly 1 at each step.

To encode a Type I instruction p_i on c_1, (resp. c_2), we start from point $(c_1, c_2, 0)$ in location P_i, define the flow in P_i to be $\dot{\mathbf{x}} = (1, 0, 1)$ (resp. $\dot{\mathbf{x}} = (0, 1, 1)$) and the guard to be $z = 1$, jump to T_i. Then in P_i the value of counter c_1 (resp. c_2) is increased by 1. In T_i we define the flow $\dot{\mathbf{x}} = (0, 0, -1)$ and guard $z = 0$ to reset the timer z to 0 and jump to P_j.

For a Type II instruction when $k = 1$, the flow in P_i is defined as $\dot{\mathbf{x}} = (-1, 0, 1)$ with guards: (1) $x = 0 \wedge z < 1$, jump to P_{j_2}; (2) $z = 1$, jump to T_i. In this case, we immediately test whether $x = 0$ when entering P_i and jump to P_{j_2} if it is true. Otherwise, for one time unit we apply derivative $(-1, 0, 1)$, which decreases counter c_1 by 1 (the x-coordinate) and increase the timer by 1 (the z-coordinate), at which point guard (2) is true. We then go to T_i, define the flow $\dot{\mathbf{x}} = (0, 0, -1)$ and guard $z = 0$ to reset the timer z to 0 and jump to P_{j_1}. A similar encoding can be defined when $k = 2$ mutatis mutandis.

We may assume without loss of generality that the machine only halts when both counters have value zero and the (single) halting instruction is denoted p_H. The reachability problem starts at point $(x, y, 0)$ in initial location P_0 and the problem is to determine if the 3-RHPCD ever reaches point $(0, 0, 0)$ in location P_H. Note that the defined region for the 3-RHPCD is unbounded in the x and y coordinates in all locations, since these coordinates store the counters c_1 and c_2 respectively. The number of regions is bounded. Full details are shown in Table 3.3.

Minsky machine $\mathcal{M}$	3-RHPCD	
p_i	P_i	T_i
`Add` 1 `to` c_1; `goto` p_j	support set: R flow: $\dot{\mathbf{x}} = (1, 0, 1)$ guard: $z = 1$, go to T_i	support set: R flow: $\dot{\mathbf{x}} = (0, 0, -1)$ guard: $z = 0$, go to P_j
`If` $c_1 \neq 0$ `then` $c_1 := c_1 - 1$; `goto` p_{j_1}; `else goto` p_{j_2}	support set: R flow: $\dot{\mathbf{x}} = (-1, 0, 1)$ guard: $z = 1$ go to T_i $x = 0 \wedge z < 1$, go to P_{j_2}	support set: R flow: $\dot{\mathbf{x}} = (0, 0, -1)$ guard: $z = 0$, go to P_{j_1}

Table 3.3: An unbounded 3-RHPCD simulating the Minsky machine $\mathcal{M}$ for counter c_1, where $R = [0, \infty) \times [0, \infty) \times [0, 1]$

As any configuration (p_i, c_1, c_2) of $\mathcal{M}$ including the initial point is encoded by the point $(c_1, c_2, 0)$ in location P_i in the 3-RHPCD, the halting configuration $(p_H, 0, 0)$ of $\mathcal{M}$ is encoded by the point $(0, 0, 0)$ in location P_H in the 3-RHPCD, and a one-step computation from (p_i, c_1, c_2) to (p_j, c_1', c_2') in $\mathcal{M}$ is encoded by the trajectory segment from point $(c_1, c_2, 0)$ in location P_i to the point $(c_1', c_2', 0)$ in location P_j, thus a 3-RHPCD can simulate a two counter machine and the reachability problem for a 3-RHPCD is undecidable.

□

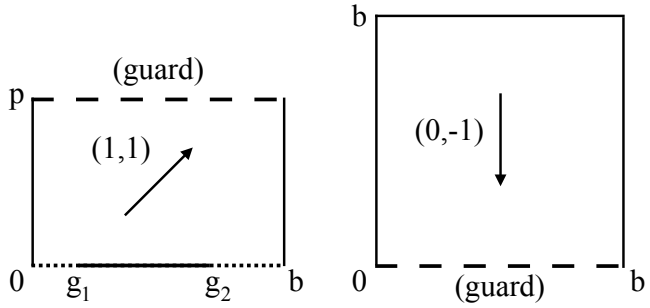
We now show a lower bound for *nondeterministic* 2-RHPCDs. A nondeterministic RHPCD can potentially have more than one possible discrete transition available within a location.

Corollary 4. *The reachability problem for bounded nondeterministic 2-RHPCDs is PSPACE-hard.*

Proof. It was shown in [32] that the reachability (i.e. halting) problem for a nondeterministic bounded 1-counter machine $\mathcal{M}$ is PSPACE-complete when the value

of the counter is bounded by a constant $c \in \mathbb{N}$ and when the machine may add or subtract an arbitrary constant $p \in [0, c]$ to the counter in each transition. Transitions are endowed with guards, which are intervals $[g_1, g_2]$ with $0 \leq g_1 \leq g_2 \leq c$, defining that a transition may be taken when the counter lies within the interval. An instruction k, defining a transition between locations p_i and p_j is written in the form $k = (p_i, p, g_1, g_2, p_j)$. See [32] for full details.

Theorem 12 shows a simulation of an (unbounded) 2-counter machine by an unbounded 3-RHPCD, where the x and y coordinates store the values of the two counters c_1 and c_2 (respectively) and the z coordinate is a timer, bounded in the interval $[0, 1]$ and used to add/subtract one from a counter. We use a similar construction in dimension two to simulate $\mathcal{M}$. The x coordinate is used to store the counter and the y coordinate is used as the timer to add or subtract an arbitrary amount from $[0, c]$ to the counter in each location. To simulate an



(a) Location P_k (b) Location T_k

Figure 3.9: Nondeterministic 2-RHPCD simulating bounded 1-counter machine

instruction $k = (p_i, p, g_1, g_2, p_j)$, we first define a location P_k. See Figure 3.9. Let $I = [0, c] \times (0, c]$ and then define the invariant of P_k to be $I \cup ([g_1, g_2] \times \{0\})$, thus P_k is only defined when the y coordinate is positive, or equal to 0 with the x coordinate in $[g_1, g_2]$. The derivative of P_k is $(1, 1)$ if $p > 0$ or else $(-1, 1)$ and the transition guard to location T_k is defined at $[0, c] \times \{p\}$ (we thus remove $[0, c] \times \{p\}$ from the invariant of the location since they should not overlap). Therefore, starting from a point $(g, 0)$ in location P_k, where $g \in [g_1, g_2]$, the trajectory hits the guard at point $(g \pm p, p)$, depending on whether we added or subtracted p.

T_k works as in Theorem 12 to zero the timer (y coordinate), with derivative $(0, -1)$ and invariant $[0, b] \times (0, b]$. Thus configuration $(g, 0)$ in P_k will reach point

$(g \pm p, 0)$ in location T_k. The transition guard of T_k is defined at $[0, b] \times \{0\}$ and nondeterministically transitions to any location $P_{k'}$ where k' is an instruction of the form $(l_j, p', g'_1, g'_2, l'_j)$ for some p', g'_1, g'_2, l'_j. Note that we define the invariant of T_k to be $[0, b] \times (0, b]$ so that T_k can be used in all simulations of instructions k which have the same target location l_j, hence the number of locations required is reduced. However, we need one location P_k for one instruction k.

The initial configuration of the 2-RHPCD is point $(0, 0)$ in location P_1. Determining if $\mathcal{M}$ ever reaches the halting state $(l_H, 0)$ is PSPACE-complete, which proves the PSPACE-hardness of reaching point $(0, 0)$ in location P_H of the 2-RHPCD since the above construction simulates the operations of machine $\mathcal{M}$ when started in the initial configuration.

Clearly the description size of the 2-RHPCD is polynomial is the size of $\mathcal{M}$. The initial configuration of the 2-RHPCD is point $(0, 0)$ in location P_1. Determining if $\mathcal{M}$ ever reaches the halting state p_H with counter 0 is PSPACE-complete, which proves the PSPACE-hardness of reaching point $(0, 0)$ in location P_H of the 2-RHPCD since the above construction simulates the operations of machine $\mathcal{M}$ when started in the initial configuration and the reduction from $\mathcal{M}$ is polynomial time. □

Corollary 5. *The reachability problem for bounded nondeterministic 2-RHPCDs is PSPACE-complete.*

Proof. Proposition 1 can clearly be seen to still hold even when the system is nondeterministic, since the description size of the number of configurations is still bounded by a polynomial. Thus, by Proposition 1 and Corollary 4 the corollary holds. □

3.5 Summary of Chapter

In this chapter we first showed a 2-RHPCD with arbitrary constant flows can simulate a bounded 1-PAM. Then together with some known results, we know a 2-RHPCD with any one of the three computational powers: arbitrary constant flows, comparative guards and affine resets, is equivalent to a bounded 1-PAM with respect to the reachability problem. Then we showed the reachability problem for

n-RHPCDs is in SPACE, and in the 3-dimensional case it is co-NP-hard to decide. When we add nondeterminism to the model, the problem for the nondeterministic 2-RHPCD becomes PSPACE-complete. If we allowed unboundedness, we can show the problem is undecidable. We also showed the reachability problem for 1-PAM is co-NP-hard and for 1-PRM is undecidable. See the list below.

- Reachability problem for
  - bounded 2-HPCDs (1-PAM equivalent);
  - nondeterministic 2-RHPCDs (PSPACE-complete);
  - 3-RHPCDs (co-NP-hard);
  - unbounded 3-RHPCDs (undecidable);
  - n-RHPCDs (in PSPACE);
  - 1-PAM (co-NP-hard);
  - 1-PRM (undecidable);

The questions that remain to be answered are: whether the reachability problem for 2-HPCD is decidable; whether the same problem for 2-RHPCD is in NP; whether the reachability for 3-RHPCD is in NP, or in PSPACE?

Chapter 4

Mortality Problems for HPCDs

In this chapter we shall explore the mortality problem for variant HPCD systems. As we stated earlier, the mortality problem can be seen as a variation of the reachability problem. From this chapter the readers will see that, on one hand, most results we showed in Chapter 3 also hold for mortality; on the other hand, as mortality deals with all the trajectories, in most cases the proof of mortality seems a bit more difficult than the case of reachability.

We first show mortality for bounded 3-RHPCD is also co-NP-hard, by a similar idea of encoding simultaneous incongruences problem. Then instead of the n-dimensional case of reachability, we show mortality problem for bounded 2-RHPCD is in PSPACE. When extended to unbounded case, mortality for 3-RHPCD is also undecidable by encoding a halting problem for Minsky machine.

Finally, we shall discuss about a similar but more general property for hybrid systems compared to mortality called *stability*. Unlike mortality dealing with the behaviours of all trajectories in finite time, stability is a property studying infinite time. As stability is only studied in Section 4.3, we given the related definitions within the section instead of in the preliminaries part.

4.1 Higher dimensional RHPCDs

Theorem 13. *The mortality problem for bounded* 3*-RHPCDs is co-NP-hard.*

Proof. We use a similar approach as in the proof of Theorem 9. The notations used in this proof are the same defined as in the proof of Theorem 9 unless otherwise stated. We encode an instance of the simultaneous incongruences problem into a 3-RPHCD. We construct our 3-RHPCD in such a way that the system is mortal if and only if there is no solution for the corresponding simultaneous incongruences problem, otherwise the system is immortal.

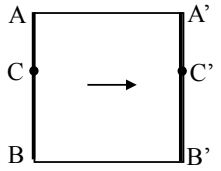
For a pair (a_i, b_i) in the simultaneous incongruences problem, the derivatives of the associated regions $\overline{A}_i$ and $\overline{B}_i$ in locations P and Q are defined the same as in the proof of Theorem 9. In contrast to Theorem 9, in the mortality problem, we are not only concerned about some trajectories starting from certain points $(0, 0, k), 0 < k \leq \rho$, but want to know whether *all* the trajectories halt.

In the following part we assume i is odd, similar analysis can be applied to the case when i is even. According to the flow $\dot{\mathbf{x}} = (1, 1, -1)$ of an odd region $\overline{A}_i$ in location P, there are 2 boundaries the trajectories will eventually reach: the O surface and the $y = \rho$ surface (some trajectories may also reach the $\overline{X}_{i+}$ or $\overline{X}_{i-}$ surface, but they will jump to location Q and jump back, then reach either one of the above two surfaces at the end). In odd $\overline{A}_i$ in P, all the trajectories which reach the $y = \rho$ surface or reach the strip G_i on the O surface are considered as mortal trajectories and will jump to location M, in which all the trajectories will eventually halt. The trajectories which reach the O surfaces but do not reach the strip G_i are considered as the potential solution trajectories and move on by following the flows for a further check.

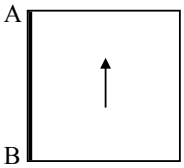
In contrast to the proof of Theorem 9, in region $\overline{A}_n$ (or $\overline{B}_n$ depending on whether i is odd or even) if any trajectory reaches the surface O but does not reach the strip G_n, we do not conclude that we find a solution k. Instead, we keep moving in P until we reach the guard, jump to location T, reset the trajectory to the point $(0, 0, k)$ and go to location P to start the test again. If k indeed is a correct solution to the corresponding simultaneous incongruences problem, the system will loop forever; otherwise the trajectory will go to location M at some

Location	Support Set	Flows	Guards
P	$\overline{A} \cup \overline{B}$	$\overline{A}_i$ (i is odd): $(1, 1, -1)$ $A_i \cup F_{i+}$ (i is even): $(0, -1, 1)$ $B_i \cup F_{i-}$ (i is odd): $(0, -1, -1)$ $\overline{B}_i$ (i is even): $(1, 1, 1)$	$\overline{X}_{i+}$ $(i = 1, 3, ..., n-1)$, $\overline{X}_{i-}$ $(i = 2, 4, ...n)$, F_{i+} $(i = 2, 4, ..., n)$, F_{i-} $(i = 1, 3, ..., n-1)$: jump to Q
			$y = \rho, G_i$: jump to M
Q	$\overline{A} \cup \overline{B}$	$\overline{A}_i$ (i is odd): $(-1, 0, 0)$ $A_i \cup F_{i+}$ (i is even): $(1, 0, 0)$ $B_i \cup F_{i-}$ (i is odd): $(1, 0, 0)$ $\overline{B}_i$ (i is even): $(-1, 0, 0)$	$\overline{X}_{i+}$ $(i = 0, 2, ..., n-2)$, $\overline{X}_{i-}$ $(i = 1, 3, ..., n-1)$: jump to P
			$\overline{X}_{n+}$: jump to T
T	$\overline{A} \cup \overline{B}$	$(-1, 0, 0)$	$x = 0$: jump to P
M	$\overline{A} \cup \overline{B}$	$(-1, 0, 0)$	None

Table 4.1: Mortality problem for 3RHPCD



(a) AB is mapped onto $A'B'$



(b) AB "collapses" to A

Figure 4.1: Edge-to-edge and edge-to-point mappings

region odd $\overline{A}_i$ or even $\overline{B}_i$ in location P. In location M, we have no outgoing transitions and follow derivative $(-1, 0, 0)$. Since the support set is bounded, any trajectory which reaches M will thus eventually halt. Full details are shown in Table 4.1. □

Proposition 2. *The mortality problem for bounded 2-RHPCDs is in PSPACE.*

Proof. Given an n-RHPCD $\mathcal{H}$, using a similar approach as in the proof of Proposition 1, we can show that the mortality problem for 2-RHPCDs is also in PSPACE. According to the constants in the description of $\mathcal{H}$, we can use a similar method as in the reachability proof to find a γ' which allows us to define a new 2-RHPCD $\mathcal{H}''$ such that $\mathcal{H}''$ is mortal iff $\mathcal{H}$ is mortal, where $\mathcal{H}''$ is described by integer coefficients. From the reachability result above we know it is possible to enumerate every integer configuration of $\mathcal{H}''$, as $\mathcal{H}''$ is bounded, and check whether every trajectory halts starting from integer configuration of $\mathcal{H}''$ in PSPACE.

Intuitively, if we connect every adjacent integer point in a 2-PCD (a location) of $\mathcal{H}''$, then each 2-PCD is tessellated by squares of length 1 and the corner points of all squares are integer points since they are the integer configurations of $\mathcal{H}''$. Also each square has exactly one dynamic vector where $\dot{x}_1, \dot{x}_2 \in \{0, 1, -1\}$. We name this technique a *rectilinear tessellation*. An edge of a square will either be mapped onto another edge, or "collapse" to a single point. See Figure 4.1 for example. In the first case, the local coordinates of the points on an edge are preserved after the mapping. In other words, if point C is on edge AB and C' on edge $A'B'$ is the image of C after the mapping, then $\frac{|AC|}{|CB|} = \frac{|A'C'|}{|C'B'|}$. Thus all points on the same edge have the same symbolic dynamics. Hence for the mortality problem, we only need to consider the corner points of all squares (all the integer points), as well as the middle points of all the edges (in the case the edge is defined by an open set and the end points does not belong to the edge), and all other points will have the same symbolic dynamics as them. To check the middle points of edges simply double the size of γ' and all the points become integers. As long as all the trajectories halt starting from the integer points and the middle points of all the edges, we can conclude that the whole system is mortal. According to the result above, clearly this can be done in PSPACE.

Note that the PSPACE result of mortality only holds for 2-RHPCD, as the local coordinates of points are not preserved in higher dimensions. □

4.2 Extensions of RHPCDs

Theorem 14. *The mortality problem is undecidable for unbounded* 3*-RHPCDs.*

Proof. It was proven in [20] that determining if a Minsky machine, $\mathcal{M}'$, is mortal (i.e. if it halts on all possible configurations) is undecidable. Our approach will be to encode such a Minsky machine $\mathcal{M}'$ using an unbounded 3-RHPCD in a similar way to the proof of Thenrem 12. The problem arises however that for mortality, we must prove that *every* initial configuration will eventually halt. We now define a variant of simulation which is required for this proof.

Previously, we defined simulation in terms of reachability, but now we use a similar notion in terms of mortality. Given a Minsky machine $\mathcal{M}'$, we say that

an HPCD $\mathcal{H}$ simulates $\mathcal{M}'$ with respect to mortality if properties (2) and (3) of Definition 5 are true, and for any configuration c of $\mathcal{H}$, the trajectory of c will, in finite time, either reach a configuration c' which is the unique encoding of a configuration $m_{c'}$ of $\mathcal{M}'$, after which $\mathcal{H}$ behaves as a simulation of $m_{c'}$, or else halt before reaching such a configuration. Note under this definition that we do not have an initial configuration of $\mathcal{M}'$ or $\mathcal{H}$. Thus $\mathcal{H}$ is mortal if and only if $\mathcal{M}'$ is.

If there exists an immortal run of the machine $\mathcal{M}'$, then there also exists an infinite trajectory of the 3-RHPCD by the above proof. Assume by contradiction that machine $\mathcal{M}'$ is mortal but there exists an infinite trajectory of the 3-RHPCD. We will deal with points not reaching the halting location first.

Assume first that such a trajectory starts in a location P_i or T_i where i is not the halting instruction. Then, by the construction, after a finite number of transitions, the system will reach some location P_j at point $(x, y, 0)$. Assuming that $x, y > 1$, then clearly $(x, y, 0)$ starting in location P_j has a similar dynamics as $(\lfloor x \rfloor, \lfloor y \rfloor, 0)$ starting in location P_j until either $x < 1$ or $y < 1$. This is because the length of time between transitions will always be 1 until this point by the use of timer z and the derivatives of all variables always come from $\{0, \pm 1\}$.

We now deal with the case where $x < 1$ or $y < 1$ at some point, corresponding to a counter being (almost) zero. We slightly modify the 3-RHPCD so that for Type II instructions, if the second guard is true ($x = 0 \wedge z < 1$), we will first zero the z-coordinate (using a new location similar to T_i), before transitioning to P_{j_2}. This means that after such a transition, z, as well as one or both of x, y will be zero. This means that again, $(x, y, 0)$ behaves the same as $(\lfloor x \rfloor, \lfloor y \rfloor, 0)$ in this location. Therefore, any initial configuration corresponds to some initial configuration of $\mathcal{M}'$ and therefore will eventually have zero in both counters and jump to the halting instruction which we define next.

In a similar way to the proof of Theorem 13, we define a 'mortal location' l_M. We define the invariant of l_M as the cube $[0, 1) \times [0, 1) \times [0, 1)$ and the derivative of this cube to be $(-1, -1, -1)$; thus any trajectory reaching l_M halts. Since a correct encoding of $\mathcal{M}'$ will only reach the halting state if both counters are zero, we see that $(0, 0, 0)$ in location l_M is the unique encoding of the halting configuration of $\mathcal{M}'$. □

4.3 Stability for HPCDs

In this section we will discuss about a property similar to mortality but in a more general sense called *stability*. Stability is one of the key properties of dynamical systems and has been widely studied. Intuitively, a dynamical system is stable means every trajectory starting near to some equilibrium point (formally defined below) will stay close to it (Lyapunov stability) and converge to that point in the limit (asymptotic stability). In [58] it was proven that Lyapunov and asymptotic stability are undecidable for PCD in 5 dimensions. Here we adapt their method to show Lyapunov and asymptotic stability for bounded 4-HPCD systems are decidable if and only if reachability for 1-PAM is decidable, based on the fact that reachability for 1-PAM is equivalent to reachability for bounded 2-HPCD. We start with some definitions:

Definition 13. [**Equilibrium Point**] *An* equilibrium point *of an HPCD system is a point* $\mathbf{0}$ *such that any trajectory starting at* $\mathbf{0}$ *remains at* $\mathbf{0}$.

Without loss of generality, we set the equilibrium point to be the origin. Also, we define the equilibrium point with respect to the continuous part and regardless of the discrete locations. This is because for stability we care more about the property of the continuous part, and an HPCD system changing its locations can also be seen as a PCD system changing underlying dynamics. Now we define two types of stability:

Definition 14. [**Lyapunov Stability**] *An HPCD system is said to be* Lyapunov stable *if for every* $\varepsilon > 0$*, there exists a* $\delta > 0$ *such that for every trajectory* ξ *with* $\xi(0) \in B_{\mathbf{0}}(\delta)$*, where* $B_{\mathbf{0}}(\delta)$ *origin-centered open ball of radius* δ *of appropriate dimension, then* $\xi(t) \in B_{\mathbf{0}}(\varepsilon)$ *holds for every* $t > 0$.

Definition 15. [**Asymptotic Stability**] *An HPCD system is called* asymptotically stable *if it is Lyapunov stable and there exists a* $\delta > 0$ *such that every infinite trajectory* ξ *with* $\xi(0) \in B_{\mathbf{0}}(\delta)$ *converges to* $\mathbf{0}$.

We now following the construction in [58] to proving the following result:

Theorem 15. *Lyapunov and asymptotic stability for bounded* 4*-HPCD systems are decidable if and only if reachability for* 1*-PAM is decidable.*

Proof. From previous results ([5], Theorem 6) we know for a given 1-PAM, we can construct a bounded 2-HPCD the reachability for which is equivalent. Also we can let the corresponding final point to be the halting configuration of the 2-HPCD, so starting from the initial configuration the computation halts (i.e. the trajectory is finite) only if the final point in the 1-PAM can be reached, otherwise the computation keeps running (i.e. the trajectory is infinite).

To prove this result, we will construct a bounded 4-HPCD for the corresponding bounded 2-HPCD and reduce the reachability problem in the 2-dimensional case to the stability problem in 4 dimensions. We denote the 2-HPCD and 4-HPCD by H_2 and H_4, respectively. Also as shown in Section 3.1, for reachability a 2-RHPCD with any one of the three computational powers: affine resets, comparative guards and arbitrary constant flows has the same computational power, so without loss of generality, we assume the given bounded 2-HPCD has comparative guards, elementary flows and has no resets.

For a given H_2 with initial point (x_0, y_0) and final point (x_f, y_f), we first extend it to a 3-dimensional HPCD H_3. We know the guards of H_2 are of the form $ax + by \prec c$, where $a, b \in \mathbb{R}$ and $\prec \in \{<, \leq, =\}$. We will define the guards of the corresponding H_3 as $ax + by \prec cz$ where z is the third variable, and any flow (p, q) in H_2 will be extended to $(p, q, 0)$ with the third dimension being 0. Intuitively, H_3 is constructed by stacking uncountable many copies of a scaled H_2 in the third dimension. So if point (x_f, y_f) is reachable from point (x_0, y_0) in time t in H_2, then (zx_f, zy_f) can be reached from (zx_0, zy_0) in time zt for any value of z.

Finally we construct H_4 by adding one more dimension working as a clock. So every flow $(p, q, 0)$ in H_3 is extended to $(p, q, 0, 1)$ and all the guards are the same as the ones in H_3, i.e., the coefficients associate with the fourth dimension is always 0. The initial partition of H_4 is an infinite set of points $\{(x_0, y_0, z, 0) | z \geq 0\}$.

Now we show the following statements are equivalent:

(1) Starting from (x_0, y_0), the trajectory will eventually reach (x_f, y_f) in H_2.

(2) H_4 is Lyapunov stable.

(3) H_4 is asymptotically stable.

(1) to (2): Assume (x_f, y_f) is reachable from (x_0, y_0) in H_2. As H_2 is bounded, there exists a ball $B_{\mathbf{0}}(d)$ containing H_2 for some $d > 0$. Thus every point (x_0, y_0, z) is contained in a dz-ball centred at point (x_0, y_0, z) in dimension 3 and hence every point $(x_0, y_0, z, 0)$ is contained in a $\sqrt{d^2z^2 + z^2}$-ball around the origin in dimension 4. Let the time for a trajectory starting from (x_0, y_0) and eventually reaching (x_f, y_f) in H_2 be bounded by some $T > 0$. Then it takes at most zT time for a trajectory starting from $(x_0, y_0, z, 0)$ in H_4 to reach the corresponding final point. Thus, any trajectory starting from $(x_0, y_0, z, 0)$ is contained in a $\sqrt{d^2z^2 + z^2 + z^2T^2}$-ball centred at origin $\mathbf{0}$. Now let $\delta < \frac{\varepsilon}{\sqrt{d^2+1+T^2}}$, it can be seen that every trajectory starting from $(x_0, y_0, \delta', 0)$ where $\delta' \leq \delta$ remains within the ε-ball. We also have all the points within the δ-ball are a subset of $\{(x_0, y_0, \delta', 0) | \delta' \leq \delta\}$. Therefore, every trajectory starting from the points within a δ-ball centred at origin is contained in an ε-ball centred at origin, which implies H_4 is Lyapunov stable.

(1) to (3): From (1) we know H_4 is Lyapunov stable, and it is easy to see there are no infinite trajectories. Hence it is also asymptotically stable.

(2) to (1): Assume H_4 is Lyapunov stable but (x_f, y_f) is not reachable from (x_0, y_0) in H_2. Then starting from the initial configuration the trajectory in H_2 will be infinite, which lead to the fourth variable in H_4 which is time growing unboundedly, and thus H_4 can not be Lyapunov stable, which is a contradiction.

(3) to (2): Directly from the definition of asymptotic stability. □

4.4 Summary of Chapter

We first showed mortality for bounded 3-RHPCD is co-NP-hard. Then we showed mortality problem for bounded 2-RHPCD is in PSPACE. When extended to the unbounded case, mortality for 3-RHPCD becomes undecidable. Finally, we discussed the stability property for HPCDs and showed both Lyapunov and asymptotic stability problems for 4-HPCD can be reduced to the reachability problem for 2-HPCD, and hence is equivalent to the reachability for 1-PAM. See the list below.

- Mortality problem for
  - 2-RHPCDs (in PSPACE);
  - 3-RHPCDs (co-NP-hard);
  - unbounded 3-RHPCDs (undecidable);
- Lyapunov and asymptotic stability problem for 4-HPCD (1-PAM equivalent);

The questions that remain to be answered are: whether the mortality problem for 2-RHPCD is in NP; whether the same problem in dimension three is PSPACE-complete; whether the mortality problem for 1-PAM is decidable, or at least NP-hard (see the discussion in Chapter 3)?

Chapter 5

Scalar Ambiguity and Freeness Problems

In this chapter we shall explore two reachability type problems, or more precisely, freeness type problems for matrix semigroups, called scalar ambiguity and scalar freeness problems.

We first show, in the general case, like many other decision problems for matrix semigroups, the undecidability for these two problems also start from dimension three or four. We use a common method for undecidability proof for matrix semigroups, encoding an instance of PCP problem (in fact we use a variation of classic PCP called MMPCP) into matrices.

We then study the above two problems over bounded languages of matrices, which means the matrices used must be in a fixed order. We are able to show the problems are still undecidable under this restriction, by a reduction from the Hilbert's tenth problem. However, the results holds only for the higher dimensional case due to the method applied.

Finally, from the connection between matrices and Probabilistic Finite Automata (PFA), we shall study a related ambiguity problem for PFA. Associated with each input word is the probability of that word being accepted by the PFA. We show that determining whether every probability is unique is undecidable over a bounded language. In other words, to determine if there exists two input words which have the same probability of being accepted is undecidable. This is a similar concept to the *threshold isolation problem* shown in [21] to be undecidable, where

we ask if each probability can be approximated arbitrarily closely.

5.1 Matrix Semigroups

We start this section by exploring the connections between the scalar ambiguity problem, the scalar freeness problem and the matrix semigroup freeness problem.

Proposition 3. *Given a semigroup of matrices $\mathcal{S}$ generated by a finite set $\mathcal{G}$, and two column vectors ρ and τ, let $\Lambda(\mathcal{G})$ be a set of scalars generated by $\mathcal{G}, \rho$ and τ. Then the following relations hold:*

(1) If $\Lambda(\mathcal{G})$ is ambiguous, then $\Lambda(\mathcal{G})$ is not free.

(2) if $\Lambda(\mathcal{G})$ is free, then $\mathcal{S}$ is free.

Proof. (1) Suppose $\Lambda(\mathcal{G})$ is ambigious, then by definition there exist two matrices $M_1, M_2 \in \mathcal{S}, M_1 \neq M_2$ such that $\rho^T M_1 \tau = \rho^T M_2 \tau$. If M_1, M_2 are different, then their factorisations must be different. Thus, there exists factorizations $M_1 = G_{i_1} G_{i_2} \dots G_{i_{m_1}} \neq G_{j_1} G_{j_2} \dots G_{j_{m_2}} = M_2$, where each $G_i, G_j \in \mathcal{G}$ and so $\Lambda(\mathcal{G})$ is not free.

(2) We proceed by contradiction. Suppose $\Lambda(\mathcal{G})$ is free but $\mathcal{S}$ is not. If $\mathcal{S}$ is not free, there exists $G_{i_1} G_{i_2} \dots G_{i_{m_1}} = G_{j_1} G_{j_2} \dots G_{j_{m_2}} \in \mathcal{S}$, where $G_i, G_j \in \mathcal{G}$, and for at least one k, $G_{i_k} \neq G_{j_k}$, or $m_1 \neq m_2$. Thus, clearly it also holds that $\rho^T G_{i_1} G_{i_2} \dots G_{i_{m_1}} \tau = \rho^T G_{j_1} G_{j_2} \dots G_{j_{m_2}} \tau$, which contradicts the definition of scalar freeness. □

It can be seen that by answering the scalar freeness problem, one can 'partly' answer the scalar ambiguity problem and the matrix semigroup freeness problem. However, neither problem is a sub-problem of the other, and it seems there is no direct connection between the scalar ambiguity problem and the matrix semigroup freeness problem. We are now ready to prove the main result of this section.

Theorem 16. *The Scalar Freeness Problem is undecidable over $\mathcal{G} \subseteq \mathbb{Q}^{3\times 3}$ and the Scalar Ambiguity Problem is undecidable over $\mathcal{G}' \subseteq \mathbb{Q}^{4\times 4}$, when $|\mathcal{G}|, |\mathcal{G}'| \geq 16$.*

Proof. We prove the result by encoding an instance of the MMPCP problem. The basic idea is inspired by [27]. We start by showing the undecidability of the scalar

freeness problem. We construct a finite set of matrices $\mathcal{G}$, generating a matrix semigroup $\mathcal{S}$ and two fixed vectors ρ and τ such that the encoded MMPCP instance has a solution if and only if the scalar set $\Lambda(\mathcal{G})$ is not free. In other words, there exists a scalar $\lambda \in \Lambda(\mathcal{G})$ such that $\lambda = \rho^T G_{i_1} G_{i_2} \dots G_{i_{m_1}} \tau = \rho^T G_{j_1} G_{j_2} \dots G_{j_{m_2}} \tau$, where $G_i, G_j \in \mathcal{G}$ and some $G_{i_k} \neq G_{j_k}$ or $m_1 \neq m_2$.

Let $\Sigma = \{x_1, x_2, \dots, x_{n-2}\}$ and $\Delta = \{x_{n-1}, x_n\}$ be distinct alphabets and $h, g : \Sigma^* \to \Delta^*$ be an instance of the mixed modification PCP. The naming convention will become apparent below. We define two injective mappings $\alpha, \beta : (\Sigma \cup \Delta)^* \to \mathbb{Q}$ by:

$$\alpha(x_{i_1} x_{i_2} \cdots x_{i_m}) = \Sigma_{j=1}^{m} i_j (n+1)^{m-j},$$
$$\beta(x_{i_1} x_{i_2} \cdots x_{i_m}) = \Sigma_{j=1}^{m} i_j (n+1)^{j-m-1},$$

and $\alpha(\varepsilon) = \beta(\varepsilon) = 0$. Thus α represents $x_{i_1} x_{i_2} \cdots x_{i_m}$ as an $(n+1)$-adic number and β represents $x_{i_1} x_{i_2} \cdots x_{i_m}$ as a fractional number $(0.x_{i_m} \cdots x_{i_2} x_{i_1})_{(n+1)}$ (e.g. if $n = 9$, then $x_1 x_2 x_3$ is represented as $\alpha(x_1 x_2 x_3) = 123_{10}$ and $\beta(x_1 x_2 x_3) = 0.321_{10}$, where subscript 10 denotes base 10). Note that $\forall w \in (\Sigma \cup \Delta)^*, \alpha(w) \in \mathbb{N}$ and $\beta(w) \in (0,1) \cap \mathbb{Q}$. It is not difficult to see that $\forall w_1, w_2 \in (\Sigma \cup \Delta)^*, (n+1)^{|w_2|}\alpha(w_1) + \alpha(w_2) = \alpha(w_1 w_2)$ and $(n+1)^{-|w_2|}\beta(w_1) + \beta(w_2) = \beta(w_1 w_2)$.

Define $\gamma' : (\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^* \to \mathbb{Q}^{3\times 3}$ by

$$\gamma'(u, v) = \begin{pmatrix} (n+1)^{|u|} & 0 & \alpha(u) \\ 0 & (n+1)^{-|v|} & \beta(v) \\ 0 & 0 & 1 \end{pmatrix}.$$

It is easy to verify that $\gamma'(u_1, v_1)\gamma'(u_2, v_2) = \gamma'(u_1 u_2, v_1 v_2)$, i.e., γ' is a homomorphism. Define two more matrices T and T^{-1} :

$$T = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad T^{-1} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We now define $\gamma : (\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^* \to \mathbb{Q}^{3\times 3}$:

$$\gamma(u,v) = T\gamma'(u,v)T^{-1} = \begin{pmatrix} (n+1)^{|u|} & (n+1)^{-|v|} - (n+1)^{|u|} & \alpha(u) + \beta(v) \\ 0 & (n+1)^{-|v|} & \beta(v) \\ 0 & 0 & 1 \end{pmatrix}.$$

We can now verify that, $\gamma(u_1,v_1)\gamma(u_2,v_2) = T\gamma'(u_1,v_1)TT^{-1}\gamma'(u_2,v_2)T^{-1} = T\gamma'(u_1u_2, v_1v_2)T^{-1} = \gamma(u_1u_2, v_1v_2)$, hence γ is a homomorphism.

Let $\mathcal{G} = \{\gamma(x_i, g(x_i)), \gamma(x_i, h(x_i)) | x_i \in \Sigma, 1 \leq i \leq n-2\}$, $\mathcal{S} = \langle \mathcal{G} \rangle$, $\rho = (1,0,0)^T$ and $\tau = (0,0,1)^T$. Assume that there exist $M_1 = G_{i_1}G_{i_2}\cdots G_{i_t} \in \langle \mathcal{G} \rangle$ and $M_2 = G_{j_1}G_{j_2}\cdots G_{j_{t'}} \in \langle \mathcal{G} \rangle$ such that $t \neq t'$ or else at least one $G_{i_p} \neq G_{j_p}$ where $1 \leq p \leq t$ and $\lambda = \rho^T M_1 \tau = \rho^T M_2 \tau$. We see that:

$$\lambda = \rho^T M_1 \tau = (M_1)_{[1,3]} = \alpha(x_{i_1}x_{i_2}\cdots x_{i_t}) + \beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})),$$
$$\lambda = \rho^T M_2 \tau = (M_2)_{[1,3]} = \alpha(x_{j_1}x_{j_2}\cdots x_{j_{t'}}) + \beta(f'_1(x_{j_1})f'_2(x_{j_2})\cdots f'_{t'}(x_{j_{t'}})),$$

where each $f_i, f'_i \in \{g, h\}$. Since $\alpha(w) \in \mathbb{N}$ and $\beta(w) \in (0,1) \cap \mathbb{Q}$, $\forall w \in (\Sigma \cup \Delta)^*$, injectivity of α and β implies that if $\rho^T M_1 \tau = \rho^T M_2 \tau$, then $t = t'$ and $i_k = j_k$ for $1 \leq k \leq t$. Furthermore, if $\rho^T M_1 \tau = \rho^T M_2 \tau$, we have that $\beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})) = \beta(f'_1(x_{i_1})f'_2(x_{i_2})\cdots f'_t(x_{i_t}))$ and since at least one $f_p \neq f'_p$ for $1 \leq p \leq t$ by our above assumption, then this corresponds to a correct solution to the mixed modification PCP instance (h, g). On the other hand, if there does not exist a solution to (h, g), then $\beta(f_1(x_{i_1})f_2(x_{i_2})\cdots f_t(x_{i_t})) \neq \beta(f'_1(x_{i_1})f'_2(x_{i_2})\cdots f'_t(x_{i_t}))$, and injectivity of β implies that $\rho^T M_1 \tau \neq \rho^T M_2 \tau$.

By Theorem 4, this implies that the result holds for $|\mathcal{G}| \geq 18$ since the MMPCP is undecidable over an alphabet of size 9. We now prove that the result holds for $|\mathcal{G}| \geq 16$. By Theorem 5 above, we may assume that $h, g : \Sigma^* \to \Delta^*$ is a Claus instance of the MMPCP problem, and that $|\Sigma| \geq 9$. Let then $\Sigma = \{x_1, x_2 \ldots, x_9\}$. Since h, g is a Claus instance, then any potential solution word w is of the form $w = x_1 w' x_9$, with $w' \in (\Sigma - \{x_1, x_9\})^*$. By symmetry, we may assume that $h_1 = h$ and by the proof in [35], $g_i = g$ and $h_i = h$ for all $1 \leq i \leq t$. Clearly then, one of $h(x_1)$ and $g(x_1)$ is a proper prefix of the other (assume $h(x_1)$ is a prefix of $g(x_1)$), otherwise a shorter solution must exist. Similarly one of $h(x_9)$ and $g(x_9)$ is a proper suffix of the other (assume that $g(x_9)$ is a suffix of $h(x_9)$; the opposite case

is similar). Now, we redefine $\rho'^T = \rho^T\gamma(x_1, h(x_1))$ and $\tau' = \gamma(x_9, g(x_9))\tau$. Finally we remove the matrices corresponding to $h(x_1)$ and $g(x_9)$ from $\mathcal{G}$ and redefine the matrices corresponding to $g(x_1)$ and $h(x_9)$ by $g'(x_1) = \gamma(x_1, h(x_1)^{-1}g(x_1))$ and $h'(x_9) = \gamma(x_9, h(x_9)g(x_9)^{-1})$ respectively. Since $h(x_1)$ is a proper prefix of $g(x_1)$, then $h(x_1)^{-1}g(x_1)$ is the suffix of $g(x_1)$ after removing the common prefix with $h(x_1)$ (similarly for $h(x_9)g(x_9)^{-1}$). Then, we see that

$$\begin{aligned} h_1(x_{i_1})h_2(x_{i_2})\cdots h_{t-1}(x_{i_{t-1}})h_t(x_{i_t}) &= g_1(x_{i_1})g_2(x_{i_2})\cdots g_{t-1}(x_{i_{t-1}})g_t(x_{i_t}) \\ \Leftrightarrow h(x_1)h(x_{i_2})\cdots h(x_{i_{t-1}})h(x_9) &= g(x_1)g(x_{i_2})\cdots g(x_{i_{t-1}})g(x_9) \\ \Leftrightarrow h(x_{i_2})\cdots h(x_{i_{t-1}})h'(x_9) &= g'(x_1)g(x_{i_2})\cdots g(x_{i_{t-1}}) \end{aligned}$$

This completes the proof of the scalar freeness problem for 16 rational matrices of dimension 3.

We now show the undecidability of the scalar ambiguity problem. The above encoding has the property that if some $\lambda = \rho^T M_1\tau = (M_1)_{[1,3]} = \rho^T M_2\tau = (M_2)_{[1,3]}$, then it implies that $M_1 = M_2$. If there exists a solution to the PCP instance, then some matrix $M \in \mathcal{S}$ has two distinct factorizations as above, one using morphisms from h, the other using morphisms from g (see the proof of the undecidability for Claus instances of MMPCP, [35]). We increase the dimension of γ by 1 to store an additional element. Each matrix of the form $\gamma(x_i, g(x_i)) \in \mathcal{G}$ is modified to $\gamma(x_i, g(x_i)) \oplus 3 \in \mathbb{Q}^{4\times4}$ and each matrix of the form $\gamma(x_i, h(x_i)) \in \mathcal{G}$ is modified to $\gamma(x_i, h(x_i)) \oplus 5 \in \mathbb{Q}^{4\times4}$. We modify ρ to $\rho \oplus 0$ and τ to $\tau \oplus 0$, which have an additional dimension which does not select this new element (of the form 3^t or 5^t). A solution to the MMPCP instance will now have two different factorizations, and the corresponding matrices will differ in one component. Therefore the ambiguity problem is undecidable for 16 matrices over $\mathbb{Q}^{4\times4}$. □

5.2 Matrix Semigroup over Bounded Languages

We now study the concept of scalar ambiguity and scalar freeness for a *bounded language* of matrices, showing that these problems are undecidable. We start with

the following corollary, which can be found in [16], or from the proof construction shown in [15].

Corollary 6. *[16] - Given an integer polynomial $P(n_1, n_2, \ldots, n_k)$, one can construct two vectors $\rho = (1, 0, \ldots, 0)^T \in \mathbb{N}^n$ and $\tau = (0, \ldots, 0, 1)^T \in \mathbb{N}^n$, an alphabet $\Sigma = \{x_1, x_2, \ldots, x_k\}$ and a homomorphism $\mu : \Sigma^* \to \mathbb{Z}^{n\times n}$, such that for any word of the form $w = x_1^{y_1} x_2^{y_2} \ldots x_k^{y_k} \in \Sigma^+$:*

$$\rho^T \mu(w) \tau = P(y_1, y_2, \ldots, y_k)^2,$$

and $\rho^T \mu(\varepsilon)\tau = 0$ for the empty word ε. The triple (ρ, μ, τ) is a linear representation of a $\mathbb{Z}$-regular formal power series $Z \in \mathbb{N}\langle\langle\Sigma\rangle\rangle$.

We will also require the following lemma.

Lemma 4. *Given two integer polynomials P_1 and P_2 over variables $(x_1, \ldots, x_k)$ and with integer coefficients. It is undecidable to decide whether there exist integers $(y_1, \ldots, y_k)$ such that $P_1^2(y_1, \ldots, y_k) = P_2^2(y_1, \ldots, y_k)$.*

Proof. Let $P(x_2, \ldots, x_k)$ be an instance of Hilbert's tenth problem, i.e. a polynomial with integer coefficients and variables. Define $P_1(x_1, x_2, \ldots, x_k) = (x_1^2 + 1)P$ and $P_2(x_1, x_2, \ldots, x_k) = (x_1^2 + 2)P$. Since $0 < x_1^2 + 1 < x_1^2 + 2$, we see that $P_1^2(x_1, x_2, \ldots, x_k) = P_2^2(x_1, x_2, \ldots, x_k) \Leftrightarrow P_1 = P_2 = 0$, which implies that $P(x_2, \ldots, x_k) = 0$, which is undecidable to determine. This result holds for any value of x_1 since $x_1^2 + 1 \neq x_1^2 + 2$. We will use this property in the later proof. □

Now we show the main result of this section.

Theorem 17. *The Scalar Freeness Problem over a bounded language is undecidable. In other words, given k matrices $M_1, M_2, \ldots, M_k \in \mathbb{Q}^{n\times n}$, generating bounded language $M = M_1^* M_2^* \cdots M_k^*$, and two vectors $\rho, \tau \in \mathbb{Z}^n$, it is undecidable to decide if there exist $l_1, l_2, \ldots, l_k, r_1, r_2, \ldots, r_k \in \mathbb{N}$ such that*

$$\rho^T M_1^{l_1} M_2^{l_2} \ldots M_k^{l_k} \tau = \rho^T M_1^{r_1} M_2^{r_2} \ldots M_k^{r_k} \tau,$$

where $l_j \neq r_j$ for at least one j.

Proof. We prove this theorem by 4 steps.

We will define a set of matrices $\{M_i, N_i | 0 \leq i \leq k+1\}$ for some $k+1 > 0$, which will define the bounded language of matrices

$$M = M_0^* M_1^* M_2^* \cdots M_k^* M_{k+1}^* N_0^* N_1^* N_2^* \cdots N_k^* N_{k+1}^*.$$

The matrices $\{M_i\}$ will encode a polynomial P_1 and matrices $\{N_i\}$ will encode a separate polynomial P_2. The proof will show that if $\rho^T A_1 \tau = \rho^T A_2 \tau$, where $A_1, A_2 \in M$ and A_1, A_2 have different factorizations, then we must have $A_1 = M_0^{j_0} M_1^{j_1} M_2^{j_2} \cdots M_k^{j_k} M_{k+1}^{j_{k+1}}$ and $A_2 = N_0^{j'_0} N_1^{j_1} N_2^{j_2} \cdots N_k^{j_k} N_{k+1}^{j'_{k+1}}$ (or vice versa). We will show that this implies that $P_1^2(j_1, \cdots, j_k) = P_2^2(j_1, \cdots, j_k)$, the determination of which was shown to be undecidable in Lemma 4.

Step 1. Given two integer coefficient polynomials P_1 and P_2 of same number of variables, from Corollary 6, we can construct an alphabet $\Sigma = \{x_1, x_2, \ldots, x_k\}$, two vectors $\rho' = (1, 0, \ldots, 0)^T, \tau' = (0, \ldots, 0, 1)^T \in \mathbb{N}^n$, and two homomorphisms $\mu_1, \mu_2 : \Sigma^* \to \mathbb{Z}^{n \times n}$ such that:

$$\rho'^T \mu_i(w) \tau' = \begin{cases} P_i(y_1, y_2, \ldots, y_k)^2, & \text{if} \quad w \in L \backslash \{\varepsilon\}; \\ 0, & \text{if} \quad w = \varepsilon; \end{cases}$$

where $i \in \{1, 2\}$ and L is the bounded language $L = x_1^* x_2^* \ldots x_k^* \subset \Sigma^*$.

Step 2. Given alphabets $K = \{0, 1, \ldots, k, k+1\}$ and $\Omega = K \cup \{\#, *\}$, define left and right desynchronizing morphisms l and $r : K^* \to \Omega^*$ by

$$\begin{aligned} l(0) &= \#0, \quad & l(1) &= *1, \quad & l(i) &= \#i, \quad & l(k+1) &= \#(k+1)\#, \\ r(0) &= \#0*, \quad & r(1) &= 1\#, \quad & r(i) &= i\#, \quad & r(k+1) &= (k+1)\#, \end{aligned}$$

where $2 \leq i \leq k$. In the sequel, by abuse of notation, we use l_j, r_j to represent the words derived from the morphisms $l(j), r(j), 0 \leq j \leq k+1$. We define a word $u \in \Omega^*$ as 'free' if there is a unique factorization of u over $\{l_j, r_j\}$.

Let $L' = l_0^* l_1^* \cdots l_{k+1}^* r_0^* r_1^* \cdots r_{k+1}^* \in \Omega^*$. We shall now prove that any word $u = l_0^{j_0} l_1^{j_1} \cdots l_{k+1}^{j_{k+1}} r_0^{j'_0} r_1^{j'_1} \cdots r_{k+1}^{j'_{k+1}} \in L'$ is *not* free if and only if all $j_i = 0$ or all $j'_i = 0$ where $1 \leq i \leq k$.

Note that no element of $\Gamma = \{l_t, r_t | 0 \leq t \leq (k+1)\}$ is a prefix of any other

word from the set, except for l_0 which is a prefix of r_0. Thus, $\Gamma \setminus \{l_0\}$ is a prefix code. If u does not begin with l_0 to some nonzero power, then by the definition of L', word u thus has a unique factorization.

If u has a prefix #0, but not #0∗, then the prefix only matches with l_0, not r_0 and this prefix can be extracted from u since it has only a single possible factorization. We can continue this argument iteratively, until we reach u which begins with #0∗. Thus assume that u begins with #0∗. Let $u = l_0u_1 = r_0v_1$ be the two possible factorizations. Since u_1 must start with ∗, then $u_1 = l_1u_2$. This implies that v_1 starts with symbol '1', which implies $v_1 = r_1v_2$ since r_1 is the only word with prefix 1. Now, u_2 must be of the form l_pu_3 for some $2 \leq p \leq k$. Then v_2 must be of the form r_pv_3. This matching continues iteratively, until eventually we reach $(k+1)$, at which point we must use l_{k+1} for the u-word and r_{k+1} for the v-word.

At this point we have the two factorizations $u = l_0^* l_0 l_1 l_2^{j_2} \cdots l_k^{j_k} l_{k+1} r_{k+1}^*$ and $u = l_0^* r_0 r_1 r_2^{j_2} \cdots r_k^{j_k} r_{k+1} r_{k+1}^*$ as the only possibilities. An example of this follows:

$$\begin{aligned} u = \#0 * 1\#3\#5\#(k+1)\# &= l_0 l_1 l_3 l_5 l_{k+1} &&= \#0 \cdot *1 \cdot \#3 \cdot \#5 \cdot \#(k+1)\# \\ &= r_0 r_1 r_3 r_5 r_{k+1} &&= \#0 * \cdot 1\# \cdot 3\# \cdot 5\# \cdot (k+1)\# \end{aligned}$$

Step 3. We now encode the words l_i and r_j $(0 \leq i, j \leq k+1)$ into rational numbers in the interval $(0, 1)$. For simplicity we first define a mapping $\sigma : \Omega \to X$, where $X = \{x_0, x_1, \ldots, x_{k+3}\}$ such that

$$\sigma(z) = \begin{cases} x_z & \text{if} \quad z \in \{0, 1, \ldots, k+1\}; \\ x_{k+2} & \text{if} \quad z = \#; \\ x_{k+3} & \text{if} \quad z = *. \end{cases}$$

We can extend σ to be a homomorphism $\sigma : \Omega^* \to X^*$. We then define a homomorphism $\beta : X^* \to (0, 1) \cap \mathbb{Q}$ in a similar way as in the proof of Theorem 16:

$$\beta(x_{i_1} x_{i_2} \cdots x_{i_m}) = \Sigma_{j=1}^{m} i_j (n+1)^{j-m-1},$$

and $\beta(\varepsilon) = 0$, where $n = |X| = k + 4$. Moreover, we use a similar definition as in the proof of Theorem 16 for γ, but only on a single word $v \in X^*$, such that

$\gamma : X^* \to \mathbb{Q}^{2\times 2}$:

$$\gamma(v) = \begin{pmatrix} (n+1)^{-|v|} & \beta(v) \\ 0 & 1 \end{pmatrix}.$$

It can be verified that $\gamma(v_1v_2) = \gamma(v_1)\gamma(v_2)$, and thus γ is a homomorphism.

Finally, we define $\gamma_l, \gamma_r : K^* \to \mathbb{Q}^{2\times 2}$ by $\gamma_l(i) = \gamma(\sigma(l_i))$ and $\gamma_r(i) = \gamma(\sigma(r_i))$, where $0 \le i \le k+1$. It can be seen that $\rho''^T\gamma_l\tau''$ and $\rho''^T\gamma_r\tau''$ are two homomorphisms from K^* to $(0,1)$, where $\rho'' = (1,0)^T$ and $\tau'' = (0,1)^T$, mapping the words derived from left and right desynchronizing morphisms l and r to $(0,1) \cap \mathbb{Q}$.

Step 4. In step 1 we showed how to encode an integer polynomial into a matrix. In step 2 and 3 we defined left and right desynchronizing morphisms and wrote them into matrix form. We now combine these steps together by defining a set of matrices $\{M_i, N_i\} \subset \mathbb{Q}^{(n+2)\times(n+2)}$:

$$\begin{aligned} &M_0 = I \oplus \gamma_l(0), \quad && M_i = \mu_1(x_i) \oplus \gamma_l(i), \quad && M_{k+1} = I \oplus \gamma_l(k+1), \\ &N_0 = I \oplus \gamma_r(0), \quad && N_i = \mu_2(x_i) \oplus \gamma_r(i), \quad && N_{k+1} = I \oplus \gamma_r(k+1), \end{aligned}$$

where $1 \le i \le k$, and I is the $n \times n$ identity matrix. Then we let a scalar λ be written as:

$$\begin{aligned} \lambda &= \rho^T M_0^{p_0} M_1^{p_1} \dots M_{k+1}^{p_{k+1}} N_0^{q_0} N_1^{q_1} \dots N_{k+1}^{q_{k+1}} \tau \\ &= \rho'^T \mu_1(w_1)\mu_2(w_2)\tau' + \rho''^T \gamma_l(v_1)\gamma_r(v_2)\tau'', \end{aligned}$$

where $\rho = (\rho'^T, \rho''^T)^T, \tau = (\tau'^T, \tau''^T)^T, w_1, w_2 \in L, v_1, v_2 = 0^*1^* \dots (k+1)^* \in K^*$. It can be seen that scalar λ contains two parts, one part consists of the homomorphisms μ_1, μ_2 we constructed in step 1 related to the polynomials, which is the integer part; the other part consists of the homomorphisms γ_l, γ_r we constructed in step 3 related to the desynchronizing morphisms, which is the fractional part. We now show that scalar λ is *not* free if and only if there exists some nonzero integer variables $(y_1, \dots, y_k)$ such that $P_1^2(y_1, \dots, y_k) = P_2^2(y_1, \dots, y_k)$.

If λ is not free, by definition there must be integers $p_0, \dots, p_{k+1}, q_0, \dots, q_{k+1}$ and $p'_0, \dots, p'_{k+1}, q'_0, \dots, q'_{k+1}$ such that

$$\lambda = \rho^T M_0^{p_0} \dots M_{k+1}^{p_{k+1}} N_0^{q_0} \dots N_{k+1}^{q_{k+1}} \tau = \rho^T M_0^{p'_0} \dots M_{k+1}^{p'_{k+1}} N_0^{q'_0} \dots N_{k+1}^{q'_{k+1}} \tau,$$

where $p_t \neq p'_t$ or $q_t \neq q'_t$ for at least one $0 \leq t \leq k+1$. Since the value of the fractional part of λ only depends on the desynchronizing morphisms, l, r, and the fractional parts are identical in both factorizations, from step 2 we have

$$\begin{aligned} &p_i = q'_i \text{ and } q_i = p'_j = 0, \text{ for } 1 \leq i, j \leq k, \text{ or} \\ &p_i = q'_i = 0 \text{ and } q_j = p'_j, \text{ for } 1 \leq i, j \leq k. \end{aligned}$$

We only consider the first case, the second case can be analysed in a similar way. As the integer parts of λ in both factorizations are also identical, and $M_0, M_{k+1}, N_0, N_{k+1}$ are defined in a way that the value of $p_0, p_{k+1}, q_0, q_{k+1}$ and $p'_0, p'_{k+1}, q'_0, q'_{k+1}$ do not affect the value of the integer part, we have

$$\rho'^T \mu_1^{p_1}(x_1) \dots \mu_1^{p_k}(x_k) \tau' = \rho'^T \mu_2^{p_1}(x_1) \dots \mu_2^{p_k}(x_k) \tau',$$

which implies that $P_1^2(p_1, \dots, p_k) = P_2^2(p_1, \dots, p_k)$. So $(p_1, \dots, p_k)$ is a solution.

If λ is free, we show there is no solution such that $P_1^2 = P_2^2$ by contradiction. Assume there is a nonzero solution $(y_1, \dots, y_k)$, such that $P_1^2(y_1, \dots, y_k) = P_2^2(y_1, \dots, y_k)$. From the way we construct P_1 and P_2 in Lemma 4, we know the value of y_1 can be any integer value without changing the equality. Thus it must be true that $P_1^2(1, y_2, \dots, y_k) = P_2^2(1, y_2, \dots, y_k)$, and there exists a word $w = x_1 x_2^{y_2} \dots x_k^{y_k} \in L^*$ such that

$$\rho'^T \mu_1(w) \tau' = \rho'^T \mu_2(w) \tau',$$

which implies that

$$\rho'^T \mu_1(x_1) \mu_2^{y_2}(x_2) \dots \mu_k^{y_k}(x_k) \tau' = \rho'^T \mu_1(x_1) \mu_2^{y_2}(x_2) \dots \mu_k^{y_k}(x_k) \tau'.$$

Since

$$\begin{aligned} M_i &= \mu_1(x_i) \oplus \gamma_l(i), \\ N_i &= \mu_2(x_i) \oplus \gamma_r(i), \end{aligned}$$

for $1 \leq i \leq k$, we can set $v = 0 \cdot 1 \cdot 2^{y_2} \cdots k^{y_k} \cdot (k+1)$, and scalar λ can be written

as

$$\begin{aligned}\lambda &= \rho'^T\mu_1(w)\tau' + \rho''^T\gamma_l(v)\tau'' &&= \rho^T M_0 M_1 M_2^{y_2}\cdots M_k^{y_k} M_{k+1}\tau \\ &= \rho'^T\mu_2(w)\tau' + \rho''^T\gamma_r(v)\tau'' &&= \rho^T N_0 N_1 N_2^{y_2}\cdots N_k^{y_k} N_{k+1}\tau.\end{aligned}$$

This shows that λ has two different factorizations, which is a contradiction. Thus we showed that scalar freeness problem can be reduced to the problem stated in Lemma 4, which is undecidable. □

Theorem 18. *The Scalar Ambiguity Problem over a bounded language is undecidable.*

Proof. We can use the same idea as in the proof of Theorem 16, increasing the dimension of matrices M_i, N_i constructed in the proof of Theorem 17 to store an additional word which is unique for each matrix. Vectors ρ, τ are modified with an additional zero-value dimension such that the value of scalar λ is not affected. Hence in the case $\lambda = \rho^T M_1\tau = \rho^T M_2\tau$, we must have $M_1 \neq M_2$. □

Corollary 7. *Vector ambiguity over a bounded language is undecidable.*

Proof. Immediately from Theorem 18 in the case when only one vector τ is considered. □

5.3 PFA on Bounded Languages

Finally, we show a result related to probabilistic finite automata.

Problem 13. **[PFA Ambiguity]** *Given a PFA $R = (u, \varphi, v)$ over a bounded language $L \in A^*$, do there exist two different words $w_1, w_2 \in L$ such that $u^T\varphi(w_1)v = u^T\varphi(w_2)v$?*

Corollary 8. *Ambiguity for PFA over a bounded language is undecidable.*

Proof. In this proof, we will construct a PFA (u, φ, v) over a bounded language L on an alphabet A. We will show that the problem to decide if there exist two different words $w_1, w_2 \in L$ such that $u^T\varphi(w_1)v = u^T\varphi(w_2)v$, can be reduced to the scalar freeness problem and hence is undecidable. The proof uses a modification of the construction in [61]; see also [16, 42].

Define $\{M'_i, N'_i | 0 \leq i \leq k+1\} \subseteq \mathbb{Z}^{(t-3)\times(t-3)}$ and $\rho', \tau' \in \mathbb{Z}^{t-3}$ to be the extended integer version of the matrices $\{M_i, N_i | 0 \leq i \leq k+1\}$ and vectors ρ, τ defined in the proof of Theorem 17, where $t > 3$ is the appropriate dimension. We increase the dimension of each M'_i, N'_i and ρ', τ' by one by defining $M''_i = tM'_i \oplus 1, N''_i = tN'_i \oplus 1$, for each $0 \leq i \leq k+1$ and $\rho'' = \rho' \oplus 1, \tau'' = \tau' \oplus 1$.

Define the morphism $\zeta : A = \{a_0, a_1, \ldots, a_{2k+3}\} \to \{M''_i, N''_i\}$ by

$$\zeta(a_j) = \begin{cases} M''_j & \text{if} \quad 0 \leq j \leq k+1; \\ N''_{j-(k+2)} & \text{if} \quad k+2 \leq j \leq 2k+3. \end{cases}$$

Then for a word $w \in A^*$, we have

$$\rho''^T \zeta(w) \tau'' = t^{|w|} \rho'^T X'_w \tau' + 1 = t^{|w|}\lambda + 1,$$

where X'_w is the matrix generated by M'_i, N'_i according to the word w and $\lambda = \rho'^T X'_w \tau' \in \mathbb{Z}$.

We then extend the dimension of the matrix $\zeta(a_j)$ to t by defining $\zeta' \to \mathbb{Z}^{t\times t}$:

$$\zeta'(a_j) = \begin{pmatrix} 0 & 0 & 0 \\ p_j & \zeta(a_j) & 0 \\ r_j & q_j^T & 0 \end{pmatrix},$$

where $p_j, q_j \in \mathbb{Z}^{(t-2)}$ and $r_j \in \mathbb{Z}$ are chosen such that, for each $\zeta'(a_j)$, the row and column sums of $\zeta'(a_j)$ are all 0 (note that these values are well defined and unique).

We now modify $\zeta'(a_j)$ so that every entry is positive. To do this we let Δ be the matrix of dimension t with all elements being 1. Let $c \in \mathbb{Z}^+$ be chosen so that $\zeta'(a_j) + c\Delta$ is a strictly positive matrix for all $1 \leq j \leq 2k+3$, and define $\hat{\zeta} : A^* \to \mathbb{Z}_+^{t\times t}$ as

$$\hat{\zeta}(a_j) = \zeta'(a_j) + c\Delta \in \mathbb{N}_{>0}^{t\times t}.$$

Finally, let $\varphi : A^* \to [0,1]^{t\times t}$ be

$$\varphi(a_j) = \frac{1}{ct}\hat{\zeta}(a_j) = \frac{1}{ct}\zeta'(a_j) + \frac{1}{t}\Delta.$$

Since row and column sums of $\zeta'(a_j)$ are all 0, and Δ is a matrix of dimension t with all elements being 1, it can be verified that all $\varphi(a_j)$ are stochastic matrices.

Then let $u = (0, \frac{1}{3}\rho''^T, 0)^T$ and $v = (0, \frac{1}{3}\tau''^T, 0)^T$, we have constructed a PFA (u, φ, v) over a bounded language $L = a_0^* a_1^* \dots a_{2k+3}^* \subseteq A^*$. Note that u, v have an L_1 norm of 1.

To see that the scalar ambiguity problem for PFA (u, φ, v) is undecidable, we note that $\Delta^n = t^{n-1}\Delta$ (as $\Delta^2 = t\Delta$), and by the definition of $\zeta'(a_j)$, it holds that $\zeta'(a_j) \cdot \Delta = \Delta \cdot \zeta'(a_j) = \mathbf{0}$ (the zero matrix). Thus,

$$\begin{aligned} u^T\varphi(w)v &= u^T\left(\left(\frac{1}{ct}\right)^{|w|}\zeta'(w) + \left(\frac{1}{t}\right)^{|w|}\Delta^{|w|}\right)v \\ &= \left(\frac{1}{ct}\right)^{|w|}(\frac{1}{9}\rho''^T\zeta(w)\tau'') + u^T\left(\frac{\Delta}{t}\right)v \qquad ;(\text{since } \Delta^{|W|} = t^{|W|-1}\Delta) \\ &= \frac{1}{9}\left(\frac{1}{ct}\right)^{|w|}(t^{|w|}\lambda + 1) + \frac{1}{t} \end{aligned}$$

Now assume there exist two different words $w_1, w_2 \in L$ such that $u^T\varphi(w_1)v = u^T\varphi(w_2)v$. Then we have

$$\frac{1}{9}\left(\frac{1}{ct}\right)^{|w_1|}(t^{|w_1|}\lambda_1 + 1) + \frac{1}{t} = \frac{1}{9}\left(\frac{1}{ct}\right)^{|w_2|}(t^{|w_2|}\lambda_2 + 1) + \frac{1}{t} \tag{5.1}$$

If $|w_1| = |w_2|$, since c and t are all fixed, we immediately get $\lambda_1 = \lambda_2$, which implies the corresponding scalar freeness problem has a solution.

If $|w_1| \neq |w_2|$, without lose of generality, we assume $|w_1| = y_1 < y_2 = |w_2|$. Then we get

$$c^{y_2-y_1}t^{y_2}\lambda_1 + (ct)^{y_2-y_1} = t^{y_2}\lambda_2 + 1,$$

But, $c^{y_2-y_1}t^{y_2}\lambda_1 + (ct)^{y_2-y_1} \mod t \equiv 0$ and $t^{y_2}\lambda_2 + 1 \mod t \equiv 1$, which gives a contradiction.

If there exist words $w_1, w_2 \in L$ such that $\rho''^T\zeta(w_1)\tau'' = \rho''^T\zeta(w_2)\tau''$ (thus the scalar freeness problem has a positive solution), then by the proof of Theorem 16, we know that $|w_1| = |w_2|$ and $\lambda_1 = \lambda_2$, therefore Equation (5.1) holds and therefore the PFA (u, φ, v) is not free. Hence the freeness problem for PFA over a bounded

language is undecidable, where the number of states of PFA is a fixed but large number depending on the instance given in Corollary 6 □

5.4 Summary of Chapter

We showed both the scalar ambiguity and freeness problems for matrix semigroups are undecidable, both in the general case or over bounded languages of matrix semigroups. However in the general case the undecidability starts in a small dimension (3 for freeness and 4 for ambiguity over rational numbers) while the bounded language case requires a large dimension. In the end we showed the ambiguity problem for a PFA over bounded languages is undecidable. See the list below.

- Scalar ambiguity problem for matrix semigroups
  - general case and over bounded languages (undecidable);
- Scalar freeness problem for matrix semigroups
  - general case and over bounded languages (undecidable);
- Ambiguity problem for PFA over bounded languages (undecidable).

The questions that remain to be answered are: in the general case, whether the scalar ambiguity and freeness problems are decidable or not in dimension two; can we reduce the dimensions required in the bounded language case for matrix semigroups?

Chapter 6

Conclusion

New results

In this thesis we studied several decision problems, or more precisely reachability type problems, for hybrid systems with linear dynamics and on matrix semigroups. Both computability and complexity results are shown. In other words, we tried to explore what types of systems for which the reachability problem and its variations are algorithmically solvable and what are not. For those problems which can be solved by algorithms, we also attempted to find the upper and lower bounds of time and memory that are required to solve the problems as tightly as we can.

The model of hybrid systems that we concentrated on in this thesis is a system called a hierarchical piecewise constant derivative (HPCD) systems. In the chapters of the Introduction and Preliminaries we explained why it is an interesting model. Firstly, the originally defined two dimensional HPCD can be seen as an intermediate model lying between two and three dimensional piecewise constant derivative (PCD) systems. It is known that the reachability for 2-PCD is decidable but the problem for 3-PCD is undecidable. So studying 2-HPCD can help to understand the boundary of decidability of reachability for PCDs. Also, the reachability for 2-HPCD is known to be equivalent to the reachability for 1-dimensional piecewise affine maps (1-PAM), a well-known discrete model for which the reachability is a longstanding open problem. So we expected the study of HPCD could help to know more about 1-PAM reachability.

In our study we first formally summarise the computability powers of HPCD

as three aspects: arbitrary constant flows, comparative guards, and affine resets (see full definitions in Section 2.1.3). We then defined a restricted model called RHPCD, and HPCD can be viewed as RHPCD endowed with these three computational powers. We started with the two dimensional case and proved in [12] that an RHPCD with arbitrary constant flows can simulate a 1-PAM. Then together with the results form [5], we know the reachability problem for an 2-RHPCD with any one of these three computational powers above is equivalent to the reachability for 1-PAM, which is a longstanding open problem. By a similar techniques we also got a branch of other related results shown in [12].

As we knew the 2-RHPCD with any one of three computational powers above lies on the boundary between decidability and undecidability for reachability, we studied further about the models on the decidable side and explored what extensions lead to undecidability. We first showed a lower bounded for 3-RHPCD that the reachability is co-NP-hard by encoding an simultaneous incongruences problem. We then showed even for the n-dimensional case, the reachability problem is decidable and can be solved in PSPACE. The proof was done based on the fact that the trajectory starting from the initial configuration will either reach the final configuration or reach a finite number of integer configurations that we can conclude the final configuration can never be reached. We also studied some extensions for RHPCDs. We found out the unboundedness allow 3-RHPCD to encode a Minsky machine hence the reachability becomes undecidable. The nondeterminism gives 2-RHPCD the power to encode a nondeterministic model called one-counter machine and the reachability is PSPACE-complete. The above results were present in [13].

In this thesis we showed the method of encoding the simultaneous incongruences problem can also be used to show the reachability for 1-PAM is co-NP-hard. Though the result in [19] is stronger, which proved that the reachability for 1-PAM over integers is PSPACE-complete, we found this technique might be interesting and present it here. We also slightly improved the undecidable result about reachability for 1-dimension piecewise rational maps in [45] by encoding a generalised Collatz problem.

For HPCD systems we also considered a problem similar to reachability, called

mortality problem. The mortality problem is also studied in other areas like matrix semigroup, which is to determine whether the zero matrix is in a given matrix semigroup, or discrete dynamical systems like 1-PAM, which is to determine whether some "mortal" point can be reached starting form every point in that system. So unlike reachability, the mortality problem deals with every possible trajectories in a system. As we discussed in Section 2.2.1, there might different ways to define mortality problem for hybrid systems, and we gave our reason why it is proper to define a hybrid system being immortal if it has an infinite trajectory and mortal if all trajectories halts. Under our definition, and using a similar proving technique, we were able to show the mortality for 3-RHPCD is co-NP-hard and can be solved in PSPACE in the two dimensional case. Also the problem becomes undecidable for unbounded 3-RHPCD. These results were also present in [13]. We then studied a more general property than mortality called stability. We gave two types of stability definitions, Lyapunov stability and asymptotic stability, and proved to decide whether a bounded 4-HPCD has these two properties is equivalent to solve the reachability problem for 1-PAMs.

Finally we studied two reachability types, or more precisely, freeness like problems named scalar ambiguity and scalar freeness problems. The study of these problems were inspired by matrix semigroup freeness and vector ambiguity problems. But instead of a matrix or a vector, we care about whether there exists a unique factorizations of a scalar generated by two vectors and a matrix semigroup. We showed in [14] these two problems are undecidable in dimension three and four respectively by encoding a variant of Post's corresponding problem. Because of the nondeterministic property of PCP, it is a useful tool in proving undecidable results for matrix semigroups which also has nondeterminism. We then studied these two problems over bounded languages for matrices, in other words we require the order of matrices to be fixed. We were able to show under this restriction the problems are still undecidable, A large number of dimensions are required, however, due to the technique we used for this case is encoding an instance of a Hilbert's tenth problem. At last, as the close connection between matrices and probabilistic finite automata, we showed an undecidable result for PFA.

Open Problems

There are still a lot of work to be carried out for both hybrid systems and matrix semigroups. We shall now list some open problems. Some of them are famous and have been standing open for a long time, so they are difficult and to solve them might need some new ideas. Some others might be easier and a variant of an existing method may work. Nonetheless, all the problems are worth to be studied. By solving the easier ones we could have a better understanding of the structure of the models being studied, and may bring a breakthrough method for the more difficult ones. We start with the open problem already mentioned in Chapter 2:

Open Problem 2. [**1-PAM Reachability**] *Given a* 1*-dimensional Piecewise Affine Map* f*, an initial point* $x \in \mathbb{Q}$ *and a final point* $y \in \mathbb{Q}$*, does there exist* $t \in \mathbb{N}$*, such that* $f^t(x) = y$*?*

This may be the most important open problem in this thesis. As the motivation of the study of HPCD are based on the fact that bounded 2-HPCDs are lying on the boundary between decidability and undecidability and it is 1-PAM equivalent. So knowing the decidability of reachability for 1-PAM can giving us a much better understanding of PCDs and of course 1-PAM itself. If we may make an (educated) guess, we would say that reachability for 1-PAMs is decidable. As either form 2-HPCD or 1-PAM itself, the lack of computational powers seems to make it impossible to simulate a Minsky machine. It will not be surprising that proving the decidability (if it is true) of reachability for 1-PAM is not easy, as even for 1-PAM over integers, reachability is already PSPACE-complete [19].

Open Problem 3. [n**-RHPCD mortality**] *Is the mortality problem for* n*-RHPCD in PSPACE, or even decidable for* $n > 2$*?*

Open Problem 4. [**1-PAM mortality**] *Is the mortality problem for* 1*-PAM undecidable, or at least NP-hard?*

Either from our proofs for HPCDs, or from the discussions about 1-PAM mortality, it looks that dealing with mortality problem for HPCDs and 1-PAMs are sometimes more difficult than the reachability problem. In this thesis, some techniques work for reachability problem are not suitable for mortality problem due

to the fact that all trajectories must be considered, and the issues existing in the mortality proofs seems difficult to overcome. We wonder if we can find some new methods to solve the above problems.

Open Problem 5. [**The** 3×3 **Scalar Ambiguity**] *Is the scalar ambiguity problem problem decidable in dimension three, or even in two?*

Open Problem 6. [**The** 2×2 **Scalar Freeness**] *Is the scalar freeness problem problem decidable in dimension two?*

Like many other problems, the scalar ambiguity and freeness problems are also difficult in dimension two. A method that is able to solve these two problems are very likely to be be helpful in solving problems like membership problem, vector and scalar reachability problems and freeness problem. Also due to the technique we used, the undecidability result of scalar ambiguity requires dimension four. It is interesting to see if it can be reduced to dimension three.

Open Problem 7. [**Skolem's Problem**] *Does a given linear recurrence sequence* $u_{k+n} = a_1 u_{k+n-1} + a_2 u_{k+n-2} + \cdots + a_k u_n$ *have a zero?*

The Skolem's problem is a famous open problem and a lot of work has been devoted to it. It is known that this problem is decidable for size 4 [36]. It is well-known that Skolem's problem can be defined by a matrix and two vectors with appropriate dimensions. Thus we believe the following freeness type problem is interesting to be studied:

Open Problem 8. *Is the scalar ambiguity problem decidable over a single matrix?*

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, and Pei-Hsin Ho. *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*. Springer, 1993.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.

[5] E. Asarin, V.P. Mysore, A. Pnueli, and G. Schneider. Low dimensional hybrid systems - decidable, undecidable, don't know. *Information and Computation*, 211:138–159, 2012.

[6] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In *CONCUR'02*, volume 2421 of *LNCS*, pages 193–208. Springer, 2002.

[7] E. Asarin, G. Schneider, and S. Yovine. Algorithmic analysis of polygonal hybrid systems, part I: Reachability. *Theoretical Computer Science*, 379:231–265, 2007.

[8] A. Balluchi, L. Benvenuti, M.D. di Benedetto, C. Pinello, and A.L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: challenges and opportunities. *Proceedings of the IEEE*, 88(7):888–912, 2000.

[9] J. Banks, V. Dragan, and A. Jones. *Chaos: a mathematical introduction*. Cambridge University Press, 2003.

[10] N. Bauer, S. Kowalewski, and G. Sand. A case study: Multi product batch plant for the demonstration of control and scheduling problems. In *ADPM*, pages 969–974, Dortmund, Germany, 2000.

[11] P. C. Bell. *Computational problems in matrix semigroups*. PhD thesis, The University of Liverpool, 2007.

[12] P. C. Bell and S. Chen. Reachability problems for hierarchical piecewise constant derivative systems. In *Reachability Problems*, volume 8169 of *Lecture Notes in Computer Science*, pages 46–58, 2013.

[13] P. C. Bell, S. Chen, and L. M. Jackson. Reachability and mortality problems for restricted hierarchical piecewise constant derivatives. In *Reachability Problems'14*, volume LNCS 8762, pages 32–45, 2014.

[14] P. C. Bell, S. Chen, and L. M. Jackson. Scalar ambiguity and freeness in matrix semigroups over bounded languages. In *Language and Automata Theory and Applications: 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, volume 9618, page 493. Springer, 2016.

[15] P. C. Bell, V. Halava, T. Harju, J. Karhumäki, and I. Potapov. Matrix equations and Hilbert's tenth problem. *International Journal of Algebra and Computation*, 18:1231–1241, 2008.

[16] P. C. Bell, V. Halava, and M. Hirvensalo. Decision problems for probabilistic finite automata on bounded languages. *Fundamenta Informaticae*, 123(1):1–14, 2012.

[17] P. C. Bell and I. Potapov. On undecidability bounds for matrix decision problems. *Theoretical Computer Science*, 391(1-2):3–13, 2008.

[18] P. C. Bell and I. Potapov. Reachability problems in quaternion matrix and rotation semigroups. *Information and Computation*, 206(11):1353–1361, 2008.

[19] A. M. Ben-Amram. Mortality of iterated piecewise affine functions over the integers: Decidability and complexity. In *30th Int. Symp. on Theoretical Aspects of Comp Sci. (STACS 2013)*, volume 20, pages 514–525, 2013.

[20] V. Blondel, O. Bournez, P. Koiran, C. Papadimitriou, and J. N. Tsitsiklis. Deciding stability and mortality of piecewise affine dynamical systems. *Theoretical Computer Science*, 255(1-2):687–696, 2001.

[21] V. Blondel and V. Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory of Computing Systems*, 36:231–245, 2003.

[22] V. Blondel and J. Tsitsiklis. When is a pair of matrices mortal? *Information Processing Letters*, 63:283–286, 1997.

[23] V. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36:1249–1274, 2000.

[24] O. Bournez and M. Branicky. On the mortality problem for matrices of low dimensions. *Theory of Computing Systems*, 35(4):433–448, 2002.

[25] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2):291–345, 2004.

[26] M. S. Branicky. *Studies in hybrid systems : modeling, analysis, and control*. PhD thesis, MIT, 1995.

[27] J. Cassaigne, J. Karhumäki, and T. Harju. On the decidability of the freeness of matrix semigroups. Technical report, Turku Centre for Computer Science, 1996.

[28] C. Choffrut and J. Karhumäki. Some decision problems on integer matrices. *Informatics and Applications*, 39:125–131, 2005.

[29] J. Conway. Unpredictable iterations. In *Proceedings of the 1972 Number Theory Conference*, pages 49–52. University of Colorado, Boulder, Colorado, 1972.

[30] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[31] A. Ehrenfeucht, J. Karhumäki, and G. Rozenberg. The (generalized) post correspondence problem with lists consisting of two words is decidable. *Theoretical Computer Science*, 21(2):119–144, 1982.

[32] J. Fearnley and M. Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. In *Automata, Languages and Programming*, volume LNCS 7966, pages 212–223, 2013.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co. New York, NY, USA, 1979.

[34] C. Haase, J. Ouaknine, and J. Worrell. On the relationship between reachability problems in timed and counter automata. In *Reachability Problems*, pages 54–65. Springer, 2012.

[35] V. Halava, T. Harju, and M. Hirvensalo. Undecidability bounds for integer matrices using Claus instances. *International Journal of Foundations of Computer Science (IJFCS)*, 18,5:931–948, 2007.

[36] V. Halava, T. Harju, M. Hirvensalo, and J. Karhumäki. Skolem's problem - on the border between decidability and undecidability. In *TUCS Technical Report Number 683*, 2005.

[37] T. Harju. Post correspondence problem and small dimensional matrices. *Lecture Notes in Computer Science*, LNCS 5583:39–46, 2009.

[38] T. Henzinger, P. Kopka, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *27th ACM STOC*, pages 373–382. ACM Press, 1995.

[39] T. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1790, pages 145–159, 2000.

[40] M. Heymann, F. Lin, G. Meyer, and S. Resmerita. Analysis of zeno behaviors in a class of hybrid systems. *Automatic Control, IEEE Transactions on*, 50(3):376–383, 2005.

[41] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, New York, 1974.

[42] M. Hirvensalo. Improved undecidability results on the emptiness problem of probabilistic and quantum cut-point languages. *SOFSEM 2007: Theory and Practice of Computer Science, Lecture Notes in Computer Science, Springer*, 4362:309–319, 2007.

[43] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 312(1):113–128, 1994.

[44] O. Kurganskyy and I. Potapov. Computation in one-dimensional piecewise maps and planar pseudo-billiard systems. In *Unconventional Computation*, pages 169–175. Springer, 2005.

[45] O. Kurganskyy, I. Potapov, and F. Sancho-Caparrini. Reachability problems in low-dimensional iterative maps. *International Journal of Foundations of Computer Science*, 19(04):935–951, 2008.

[46] Oleksiy Kurganskyy and Igor Potapov. Reachability problems for pams. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 356–368. Springer, 2016.

[47] S. A. Kurtz and J. Simon. The undecidability of the generalized collatz problem. In *Theory and Applications of Models of Computation*, pages 542–553. Springer, 2007.

[48] F. Laroussinie, N. Markey, and Ph Schnoebelen. Model checking timed automata with one or two clocks. In *CONCUR 2004-Concurrency Theory*, pages 387–401. Springer, 2004.

[49] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In *CAV93*, volume LNCS 697, pages 194–209. Springer-Verlag, 1993.

[50] Yuri Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

[51] Yuri Matiyasevich and Geraud Senizergues. Decision problems for semi-thue systems with a few rules. In *Logic in Computer Science, 1996. LICS'96. Proceedings., Eleventh Annual IEEE Symposium on*, pages 523–531. IEEE, 1996.

[52] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall International, Englewood Cliffs, 1967.

[53] T. Neary. Undecidability in binary tag systems and the post correspondence problem for five pairs of words. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 30. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[54] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, pages 149–178. Springer, 1993.

[55] A. Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

[56] K. Peters and U. Parlitz. Hybrid systems forming strange billiards. *International Journal of Bifurcation and Chaos*, 13(09):2575–2588, 2003.

[57] E. Post. a variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, pages 264–268, 1946.

[58] P. Prabhakar and M. Viswanathan. On the decidability of stability of hybrid systems. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 53–62. ACM, 2013.

[59] M. Sipser. *introduction to the theory of computation*. PWS Publishing Company, 1997.

[60] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 1–9, ACM, 1973.

[61] P. Turakainen. Generalized automata and stochastic languages. *Proceedings of the American Mathematical Society*, 21:303–309, 1969.