Low Dimensional Hybrid Systems – Decidable, Undecidable, Don't Know

Eugene Asarin[a], Venkatesh P. Mysore[b,c], Amir Pnueli[**], Gerardo Schneider[*,d,e]

[a] *LIAFA, University Paris Diderot and CNRS, France*
[b] *D. E. Shaw Research, New York, U.S.A. (current)*
[c] *New York University, U.S.A.*
[d] *Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Sweden*
[e] *Department of Informatics, University of Oslo, Norway*

Dedicated to the memory of Amir Pnueli (1941–2009)

Abstract

Even though many attempts have been made to define the boundary between decidable and undecidable hybrid systems, the affair is far from being resolved. More and more low dimensional systems are being shown to be undecidable with respect to reachability, and many open problems in between are being discovered. In this paper, we present various two dimensional hybrid systems for which the reachability problem is undecidable. We show their undecidability by simulating Minsky machines. Their proximity to the decidability frontier is understood by inspecting the most parsimonious constraints necessary to make reachability over these automata decidable. We also show that for other two dimensional systems, the reachability question remains unanswered, by proving that it is as hard as the reachability problem for piecewise affine maps on the real line, which is a well known open problem.

Key words: Hybrid systems, undecidability, piece-wise affine maps, (hierarchical) piece-wise constant derivative systems

Contents

[*] Corresponding author's address: Chalmers University of Technology, Dept. of Computing Science and Engineering. S-412 96 Gothenburg, Sweden.
[**] Affiliation at the moment of submission of this paper: New York University, USA
Email addresses: asarin@liafa.jussieu.fr (Eugene Asarin), venkatesh.mysore@deshawresearch.com (Venkatesh P. Mysore), gersch@chalmers.se (Gerardo Schneider)

1. Introduction

Hybrid automata (HA) constitute a formalism for capturing the behavior of hybrid systems that have both continuous and discrete dynamics. Hybrid automata, which can have discrete transitions between states and continuous flows within states, are a class of immense computational power. *Analyzability* can be guaranteed only on a *class* satisfying certain additional constraints limiting the number of variables (dimensions) and the nature of the dynamical evolution. Two kinds of questions are addressed in the literature concerning the analysis of hybrid systems. This first one is *reachability*: "Is a certain final state (set of states) reachable from a certain initial state (set of states)?". The second line of research addresses questions like stability and attraction. In this article, we only focus on studying whether we can *always* answer *any* reachability question over *any* hybrid automaton that satisfies certain constraints, *i.e.*, on understanding when reachability becomes *decidable*.

Despite the increasing interest in discovering new decidability results for HA (relevance to *safety verification*), there is still no clear boundary between what is decidable and what is not in such systems. HA easily become undecidable for the reachability query, with only extremely stringent restrictions leading to decidability.

In this paper, we aim at answering the question "What is the simplest class of hybrid systems for which reachability is undecidable?". Conventional answers to this question have involved proving that a certain decidable class becomes undecidable, when given some additional computational power. By following this reasoning, we provide some undecidability results, as well as proofs that for many subclasses of hybrid systems, the decidability status of the reachability question is unknown. In what concerns *undecidability* proofs, we will observe that the famously undecidable "halting problem" (or state-to-state reachability) of (two-counter) Minsky machines [1] can be naturally expressed as a reachability query over configuration space. A very important application of this result is that reachability can be proven to be undecidable for any class of automata that can simulate this two counter Minsky machine. Finite one dimensional piecewise affine maps (1-dim PAMs), for which the decidability of reachability is still an open question [2, 3], play a crucial role in the results we will present in this paper, as they will be our reference model to show the *openness* of the reachability query.

We proceed by focussing on a version of Linear Hybrid Automata [4], namely the two dimensional Hierarchical Piecewise Constant Derivative (2-dim HPCD) class introduced in [5, 6]. This is an intermediate class, between decidable two dimensional Piecewise Constant Derivative systems (2-dim PCDs) [7] and its undecidable three dimensional extension 3-dim PCDs [8]. Asarin and Schneider [5] proved that 2-dim HPCDs are *equivalent* to 1-dim PAMs, thus showing that HPCD-reachability is open. Moreover, when endowed with a little additional computational power, the 2-dim HPCD class becomes undecidable. Thus, the 2-dim HPCD class (and equivalently the 1-dim PAM class) is clearly on the boundary between decidable and undecidable HA subclasses. Subsequently, Mysore and Pnueli [9] pursued the following questions: (1) Is there a class, simpler than the 2-dim HPCD class, which can be shown to be equivalent to 1-dim PAMs? (2) Are there alternative extensions of the 2-dim HPCD class which become undecidable? (3) Can an approximate reachability algorithm be developed for the undecidable classes?

This article is an extended and revised version of the papers [5] and [9]. We introduce hydrid automata with one, two or three dimensions that span the boundary between decidability and undecidability for the reachability problem. The main results of our work are summarized in Fig. 1; the details of the class definitions (and the acronyms) will be presented in the technical sections of this paper.

In section 2, we introduce the necessary background. In section 3, we show that the reachability problem for 2-dim HPCDs, PCDs on manifolds, 2-dim PCDs with translational resets and some other classes of 2-dimensional systems is as hard as the reachability problem for 1-dim PAMs. In section 4, we show

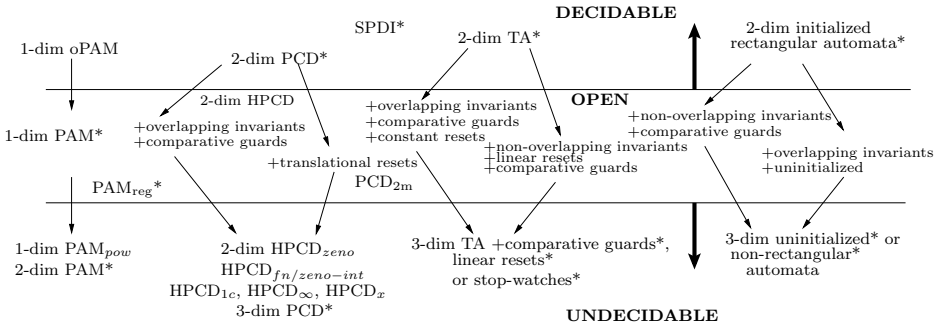


Figure 1: Decidable, open and undecidable subclasses of hybrid systems (unstarred results are contributions of this paper).

that slight extensions of the 2-dim HPCD class lead to the undecidability of the reachability question. We present a partially correct but not necessarily terminating algorithm for testing reachability in 1-dim PAMs, and show how decidable subclasses can be identified in section 5. In section 6, we present related work, and we conclude in the last section.

2. Preliminaries

In this section we define several classes of hybrid automata, two dimensional manifolds, and our reference models: PAMs and Minsky machines.

2.1. Hybrid Automata

There are many equivalent definitions of hybrid systems [10, 4, 11]. Conceptually, a hybrid automaton is a directed graph of discrete states and transitions, augmented with several real-valued continuous variables, which allows arbitrary: (1) *Invariant* expressions dictating when (for which values of variables) the system can stay in each discrete state; (2) Differential equations in the *flow* expressions in each discrete state (continuous evolution of variables with time); (3) Conditions controlling when a transition can be taken, in the *guard* expressions; (4) Equations that change the values of the variables, in the *reset* expressions during each discrete state transition (instantaneous discrete evolution). A computation of a hybrid automaton is a series of continuous evolution steps of arbitrary time-length each, interspersed with an arbitrary number of zero time-length discrete transition steps.

Definition 2.1. *An* n-dimensional hybrid automaton *is a tuple* $\mathcal{H} = (\mathcal{X}, Q, f, \mathsf{I}_0, \mathsf{Inv}, \delta)$ *where*

- $\mathcal{X} \subseteq \mathbb{R}^n$ *is the* continuous state space. *Elements of* $\mathcal{X}$ *are written as* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$*, we always use variables* $x_1, x_2, \ldots, x_n$ *to denote components of the state vector;*

- Q *is a finite set of* discrete locations*;*
- $f : Q \to (\mathcal{X} \to \mathbb{R}^n)$ *assigns a continuous vector field on* $\mathcal{X}$ *to each discrete location. While in discrete location* $\ell \in Q$*, the evolution of the continuous variables is governed by the differential equation*[1] $\dot{\mathbf{x}} = f_\ell(\mathbf{x})$*. We say that the differential equation defines the* dynamics *of location* ℓ*;*
- *The* initial condition $\mathsf{I}_0 : Q \to 2^{\mathcal{X}}$ *is a function that for each state defines the initial values of the variables of* $\mathcal{X}$*;*
- *The* invariant *or* staying *conditions* $\mathsf{Inv} : Q \to 2^{\mathcal{X}}$*,* $\mathsf{Inv}(\ell)$ *is the condition that must be satisfied by the continuous variables in order to stay in location* $\ell \in Q$*;*
- δ *is a set of transitions of the form* $tr = (\ell, g, \gamma, \ell')$ *with* $\ell, \ell' \in Q$*. Such a quadruple means that a transition from* ℓ *to* ℓ' *can be taken whenever the guard* $g \subset \mathcal{X}$ *is satisfied and then the reset relation* $\gamma \subset \mathcal{X} \times \mathcal{X}$ *is applied.* □

A hybrid automaton is said to be *deterministic* if for every location ℓ and any initial condition $\mathbf{x}_0 \in \mathsf{I}_0$, there exists at most one solution to the equation $\dot{\mathbf{x}} = f_\ell(\mathbf{x})$, and if for any location, the guards of all the outgoing discrete transitions are mutually exclusive. We consider only deterministic systems unless the contrary be specified.

A *state* is a pair $(\ell, \mathbf{x})$ consisting of a location $\ell \in Q$ and $\mathbf{x} \in \mathcal{X}$, and can change: (1) by a *discrete instantaneous* transition that changes both the location and the values of the variables according to the transition relation; or (2) by a *time delay* that changes only the values of the variables according to the dynamics of the current location. The system may stay at a location only if the invariant is true, and must take a transition before it becomes false.

A *trajectory* of a hybrid automaton $\mathcal{H}$ is a function $\Theta : [0, T] \to Q \times \mathcal{X}$, $\Theta(t) = (\ell(t), \xi(t))$ such that there exists a sequence of times $t_0 = 0 < t_1 < \ldots < t_n = T$ for which the following holds for each $1 \leq i \leq n$: (1) ℓ is constant on $[t_i, t_{i+1})$ (with value ℓ_i) and ξ is derivable on (t_i, t_{i+1}), it is right continuous and with left limits everywhere (*cadlag*); (2) There is a transition $(\ell(t_i), g, \gamma, \ell(t_{i+1})) \in \delta$ such that $\xi^-(t_{i+1}) \in g(\ell_i, \ell_{i+1})$ and $(\xi^-(t_{i+1}), \xi(t_{i+1})) \in \gamma$;[2] (3) For any $0 \leq i \leq n$, for any $t \in (t_i, t_{i+1})$, $\dot{\xi}(t) = f_{l(t)}(\xi(t))$.

2.2. Rectangular and Linear Hybrid Automata

Consider an n-dim hybrid automaton $\mathcal{H}$ with $\mathsf{V} = \{x_1, x_2, \ldots, x_n\}$ being the set of n variables. Subclasses of hybrid automata will be defined by syntactic conditions on expressions defining their ingredients (section 2.1).

A hybrid automaton $\mathcal{H}$ is *linear* (LHA) [10, 4] if: (1) The initial and invariant conditions as well as the guards and the reset relations are Boolean combinations

[1] Sometimes we will also consider differential inclusions.
[2] $\xi^-(t)$ is the left limit of ξ at t.

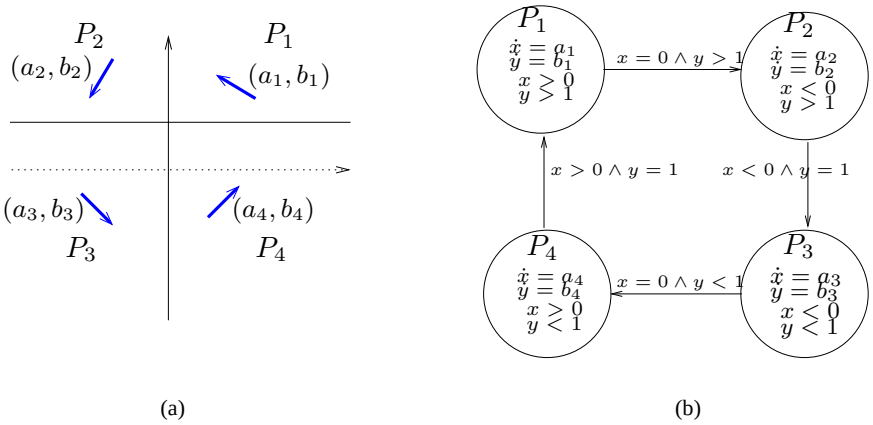


Figure 2: (a) A simple PCD; (b) Its corresponding hybrid automaton.

of linear inequalities; (2) The dynamics are defined by differential equations of the form $\dot{x} = k_x$, one for each variable $x \in \mathsf{V}$, where $k_x \in \mathbb{Q}$ is a rational constant. We say that k_x is the *slope* (or *rate*) of the variable x.

We say that a variable x is a *memory cell* if it has slope 0 in every location of $\mathcal{H}$. A variable x is a *clock* if it has slope 1 in every location.

An *n-dimensional rectangle* $R = \prod_{1 \leq i \leq n} I_i$ is the product of n intervals $I_i \subseteq \mathbb{R}$ of the real line with rational or infinite extremities.

A *rectangular constraint* refers to an expression of the form $\mathbf{x} \in R$, while a *non-rectangular* or *comparative constraint* is a conjunction of inequalities $\mathbf{x} \cdot \mathbf{c} < b$. State invariants are said to be *non-overlapping* if the regions they represent in $\mathbb{R}^n$ have disjoint interiors. A *constant reset* refers to $\mathbf{x}' = \mathbf{c}$, a *translational reset* refers to $\mathbf{x}' = \mathbf{x} + \mathbf{b}$ and an *affine reset* to $\mathbf{x}' = A\mathbf{x} + \mathbf{b}$.

A hybrid automaton is a *rectangular automaton* [4, 12, 13] if: (1) All initial conditions, invariants and guards are rectangles; (2) For each location ℓ, the dynamics has the form $\dot{\mathbf{x}} \in R_\ell$, where R_ℓ is a rectangle; (3) Reset relations are conjunctions of constraints $x_i \in I_i \wedge x_i' \in I_i'$ for some variables $x_i, i \in \rho$, and $x_i' = x_i, i \notin \rho$ for others. In an *initialized* rectangular automaton, whenever the slope of x_i changes after a transition, the variable x_i is reset (*i.e.*, $i \in \rho$).

Finally, in the *updatable timed automata* [14] parameterized by a class of updates C, all the variables are clocks (*i.e.*, their slopes equal 1), the guards and the invariants are rectangular, and the resets belong to C.

2.3. Piecewise Constant Derivative System (PCD)

We now recall PCDs, a class of hybrid systems for which the dynamics is defined by constant derivatives, such that the trajectories are continuous. A

2-dim PCD[3] is a hybrid automaton in two continuous variables, where: (1) All flow-derivatives are constants; (2) The discrete states (with their invariants) correspond to *non-overlapping* polygons in the real plane with non-empty interiors; (3) The guards correspond to boundaries between these polygons; (4) No variable can be reset during transitions, *i.e.*, γ is an identity relation.

Definition 2.2. *A* piecewise constant derivative system *(PCD) [8, 7] is a pair* $\mathcal{H} = (\mathbb{P}, \mathbb{F})$ *with* $\mathbb{P} = \{P_s\}_{s \in S}$ *a finite family of non-overlapping convex polygonal sets in* $\mathbb{R}^2$ *with non-empty interiors, and* $\mathbb{F} = \{\mathbf{c}_s\}_{s \in S}$ *a family of vectors in* $\mathbb{R}^2$*. The dynamics are determined by the equation* $\dot{\mathbf{x}} = \mathbf{c}_s$ *for* $\mathbf{x} \in P_s$*.* □

We define the *support set* of a PCD $\mathcal{H}$ to be the union of all the underlying convex polygons of the PCD, *i.e.*, $\mathsf{Supp}_{\mathsf{PCD}}(\mathcal{H}) = \cup_{s \in S} P_s$.

A well known technique for planar differential equations, and in particular for PCDs, is to replace the analysis of those systems by the analysis of edge-to-edge discrete successors [8, 15, 7] (also known as Poincaré map [16]). Given an edge e, each point on e can be represented by a local one dimensional coordinate. A one-step edge-to-edge successor in such coordinates can be written as $\mathsf{Succ}_{e'}(x) = ax + b$. In general, an n-step successor for a given sequence of edges $\sigma = e_1, e_2, \ldots, e_n$ is again a function of the above form (also see [8]).

A hybrid automaton *trajectory* in some interval $[0, T] \subseteq \mathbb{R}$, with initial condition $\mathbf{x} = \mathbf{x}_0$, can be defined as a continuous and almost-everywhere (everywhere except on a discrete set of points) derivable function $\xi(\cdot)$ such that $\xi(0) = \mathbf{x}_0$ and for all $t \in (0, T)$ if $\xi(t) \in \mathtt{int}(P_s)$[4] then $\dot{\xi}(t)$ is defined and $\dot{\xi}(t) = \mathbf{c}_s$.

Effectively, the trajectories of a PCD are restricted to be broken straight lines, with slopes changing only when a different polygonal region (new discrete state) is entered. Unlike other hybrid models, no discontinuous discrete jumps (resets) are allowed in PCDs. This constraint implies that PCD trajectories are continuous lines, which makes them more suitable for topological and geometrical analysis. The PCD restriction was motivated by one of the fundamental properties of planar systems: the evolution of a point in a plane with fixed slopes (flow) at each point, can *only* trace out a kind of contracting or expanding spiral, if not a simple finite cycle. Maler and Pnueli [7] used this property to prove that reachability is decidable for 2-dim PCDs.

Notice that PCDs can be viewed as linear hybrid automata without resets (*i.e.*, with identity reset relations), such that the locations ℓ_s correspond to regions P_s; in each location ℓ_s the invariant condition is given by the polygon itself: $\mathsf{Inv} = P_s$; the dynamics in ℓ_s is given by $\dot{\mathbf{x}} = \mathbf{c}$; and the guard associated to a transition from ℓ_s to ℓ_k is the common edge of P_s and P_k. In Fig. 2, a simple PCD and its corresponding hybrid automaton are shown.

[3] If the dimension is not explicitly mentioned, in what follows "PCD" will stand for a 2-dim PCD.

[4] $\mathtt{int}(P_s)$ is the interior of P_s.

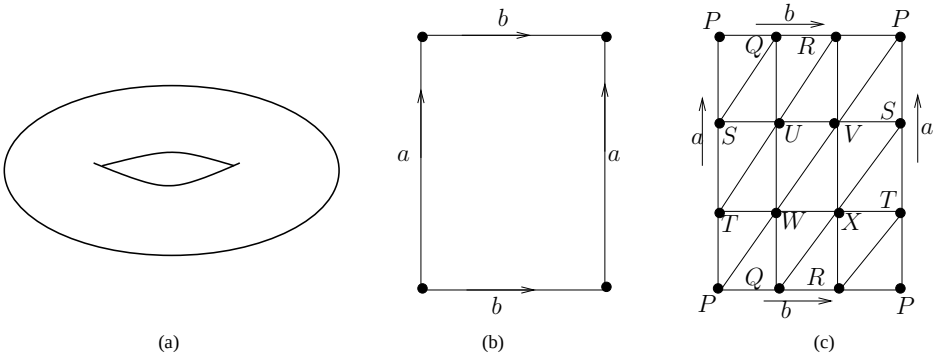


Figure 3: Representations of a Torus: (a) A surface in $\mathbb{R}^3$; (b) A square with identified edges; (c) A triangulated surface.

2.4. Two Dimensional Manifolds

All the (topological) definitions, examples and results of this section are done using the *combinatorial method* and follow [17].

A topological space is *triangulable* if obtained from a set of triangles by the identification of edges (including their vertices) and vertices, where any two triangles are identified either along a single edge or at a single vertex, or are completely disjoint. The identification should be done via an affine bijection.

Definition 2.3. *A* surface *(or* 2-dim manifold*) is a triangulable space for which in addition: (1) Each edge is identified with exactly one other edge; and (2) The triangles identified at each vertex can always be arranged in a cycle* $T_1, \ldots, T_k, T_1$ *so that adjacent triangles are identified along an edge.* □

Typical examples are the sphere, the torus (see Fig. 3) and the Klein's bottle.

A *surface with boundary* is a topological space obtained by identifying edges and vertices of a set of triangles as for surfaces except that certain edges may not be identified with another edge. These edges, which violate the definition of a surface, are called *boundary edges*, and their vertices, which also violate the definition of surface, are called *boundary vertices*. Typical examples of surfaces with boundary are the cylinder and the Möbius strip. Indeed, the cylinder is equivalent to a sphere with two disks cut out, while, a bit less intuitively, the Möbius strip can be seen as a projective plane with a disk removed.

We state now an important theorem in the topological theory of surfaces:

Theorem 2.4 ([17]).

- *Every compact, connected surface is topologically equivalent to a sphere, or a connected sum of tori, or a connected sum of projective planes.*
- *Every compact, connected surface with boundary is equivalent to either a sphere, or a connected sum of tori, or a connected sum of projective planes, in any case with some finite number of disks removed.* □

We will use this theorem without the connectedness assumption. In this case, a manifold is just a finite collection of connected manifolds described above.

2.5. Our Reference Models

In this section, we define one dimensional piecewise affine maps (PAMs) [3, 2, 18] and recall the definition of Minsky machine model of computation [19].

2.5.1. Piecewise Affine Map (PAM)

A (rational) *interval* is a subset of $\mathbb{R}$ of one of the following forms:

$$[a,b]; [a,b); (a,b]; (a,b); (-\infty,b]; (-\infty,b); [a,\infty); (a,\infty)$$

with rationals $a \leq b$.

Definition 2.5. *We say that $f : \mathbb{R} \to \mathbb{R}$ is a* one dimensional piecewise affine map *(1-dim PAM) whenever f is of the form $f(x) = a_i x + b_i$ for $x \in I_i$, where $I_i \subset \mathbb{R}$ is a finite family of disjoint (rational) intervals. The extremities of intervals I_i are referred to as* vertices *of the 1-dim PAM and their finite set is denoted V_f.* □

Note that the above definition (and related concepts) may be extended to any dimension. As we are mostly concerned with 1-dim PAMs in the rest of paper, we will write PAM to refer to 1-dim PAM, and explicitly state the dimension whenever we need to refer to higher dimensional PAMs.

Definition 2.6. *We say that a PAM is* bounded *if none of its intervals is infinite.* □

A *trajectory* of a PAM is a sequence (finite or infinite) $x_1, x_2, \ldots$ such that $x_{n+1} = f(x_n)$ for all n. We say that y is reachable from x whenever there exists a finite trajectory starting at x and arriving to y.

The reachability problem for PAMs REACH_{PAM} can be defined as:

Problem 2.7. *Given a PAM $\mathcal{A}$, is point y reachable from point x?*

Even for a function f with just two linear pieces, there is no known decision algorithm for REACH_{PAM}. The problem becomes undecidable for 2-dim PAMs, and open for dimension 1 when piecewise affine maps are replaced by polynomials [3, 2, 18]. It is also known that reachability by iteration of elementary functions on $\mathbb{R}$, such as compositions of sines and cosines, is undecidable [20].

2.5.2. Minsky Machine

We recall the definition of the Minsky machine (or, *two-counter machine*), a computational model, equivalent (with a slowdown) to a Turing machine.

Definition 2.8. *A* Minsky machine *(MM) consists of:*

- *Two unbounded registers, m, n, each containing one natural number;*

- *A list of numbered instructions.*

In a Minsky machine, only three kinds of instructions are possible:

1. Increment*: given a register and an instruction number, increment the register value, and jump to the specified instruction number;*
2. Decrement*: given a register and an instruction number, decrement the register value*[5]*, and jump to the specified instruction number;*
3. Test*: given a register and two instruction numbers, jump to the first instruction number if the register value is zero and jump to the second instruction number if the register value is positive.*
4. Halt*.* □

A configuration of a Minsky Machine is a triple (q, m, n), where q is the current instruction and m and n stand for the contents of the two counters. The halting problem over Minsky machines Halt_{MM} is a well known undecidable problem: *Given the description of a Minsky machine $\mathcal{M}$ and its initial configuration (q, m, n), will $\mathcal{M}$ eventually halt?* The reachability problem can be phrased over MM configuration space as: *Given a Minsky machine $\mathcal{M}$, will configuration (q', m', n') be reachable from configuration (q, m, n)?* The halting problem can be trivially posed as a reachability query, where the initial state q stands for the first instruction, with m and n being the allowed initial values of the counters, and the state q' represents the finite set of all Halt instruction numbers, with no restrictions on the final counter values m' and n'.

On the Notion of Simulation

Proving that two systems are *equivalent* requires showing that each *simulates* the other. Even though the idea of simulation (*abstraction* or *realization*) is accepted in the Computer Science community to mean "machines that perform the same computation" [21, 22, 19], in dynamical systems, simulation is captured by the notions of topological equivalence and homomorphism [23, 24, 25]. Defining a general notion of simulation for systems combining discrete and continuous dynamics is not easy. Some ad hoc definitions (sufficient for decidability analysis) have been presented in [26, 27, 28]; in particular the work presented in [8] provides a notion of simulation for PCDs. In the following sections, every time we establish that a system A simulates another system B in some *ad hoc* sense, we make sure that it provides a reduction from the reachability problem for A to the reachability for B; hence if reachability is undecidable for B, then it must be so for A.

3. Between Decidability and Undecidability

We show that for several natural classes of 2-dimensional hybrid systems, the reachability problem is as hard as for 1-dim PAMs, and hence, open.

[5] An attempt to decrement 0 produces an error.

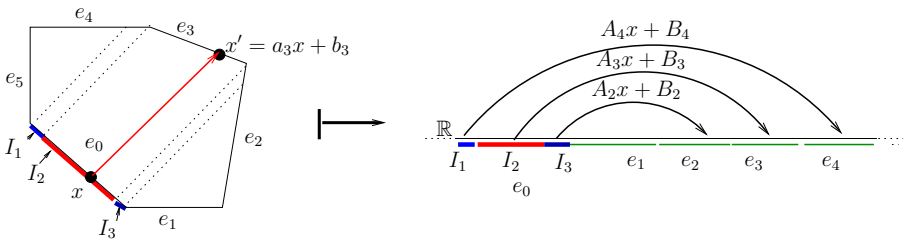


Figure 4: Sketch of the simulation of a HPCD by a PAM.

3.1. Hierarchical Piecewise Constant Derivative System (HPCD)

Hierarchical piecewise constant derivative systems can be seen as hybrid automata where the dynamics at each location is given by a PCD.

Definition 3.1. *A hierarchical piecewise constant derivative system is a hybrid automaton* $H_{PCD} = (\mathcal{X}, Q, f, \mathsf{I}_0, \mathsf{Inv}, \delta)$ *such that* Q *and* I_0 *are as before while the dynamics at each* $\ell \in Q$ *is a PCD and each transition* $tr = (\ell, g, \gamma, \ell')$ *is such that: (1) Its guard* g *is a line segment in* $\mathbb{R}^2$*; and (2) The reset relation* γ *corresponds to an affine function:* $\mathbf{x}' = \gamma(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$*. The* Inv *for a state is defined to be the support set,* $\mathsf{Supp}_{\mathsf{PCD}}(\mathcal{H})$ *minus the guards of the outgoing transitions. If all the PCDs are bounded, then* H_{PCD} *is said to be* bounded. □

Coefficients in the (in-)equations of the ingredients are assumed to be rational.

We introduce a 1-dim coordinate system on each edge e of the polygonal region in every PCD, and on every guard g of the HPCD. We denote a point with local coordinates x on edge e by (e, x), or whenever unambiguous, just as x.

It can be argued that the term *hierarchical* in the above definition is superfluous and that in fact HPCDs are equivalent to two dimensional linear hybrid automata. The definition is intended to emphasize the fact that the trajectory behaves mostly like a PCD, with a few reset induced discontinuities. The HPCD reachability problem REACH_{HPCD} can be defined as:

Problem 3.2. *Given a HPCD* $\mathcal{H}$*, is the state* $(\ell_f, \mathbf{x}_f)$ *reachable from* $(\ell_0, \mathbf{x}_0)$*?*

To prove its openness, we will show that each HPCD $\mathcal{H}$ can be simulated by a PAM $\mathcal{A}$, and that there is a HPCD $\mathcal{H}$ that simulates $\mathcal{A}$. For proving the first, we should: (1) Encode an initial and final point of $\mathcal{H}$ by points on some intervals of $\mathcal{A}$; (2) Represent a configuration of $\mathcal{H}$ by a configuration of $\mathcal{A}$; (3) Simulate an edge-to-edge transition of $\mathcal{H}$ by some function application on $\mathcal{A}$.

Lemma 3.3 (PAMs simulate HPCDs). *Every bounded HPCD* $\mathcal{H}$ *can be simulated by a PAM.*

Proof: We arrange all the edges of $\mathcal{H}$ in the real line (in an arbitrary order) and we represent each edge-to-edge successor function and each reset function by an affine map (restricted to an interval). Assembling all those affine maps together yields the PAM $\mathcal{A}$ simulating $\mathcal{H}$ (see Fig. 4).

Let $\mathcal{H}$ be a HPCD and PCD_i the PCD of location ℓ_i. We encode each region of PCD_i by parts of a PAM $\mathcal{A}$. Let e_0 be an input edge of region R and $e_1, \cdots, e_k$ be output edges of R and reachable from e_0 by the one-step successor $\mathsf{Succ}_{e_0 e_i}(\lambda) = a_i\lambda + b_i$ $(1 \leq i \leq k)$ (see Fig. 4). We partition edge e_0 into intervals $I_1, \cdots I_k$ in the following way: $I_i = \mathsf{Pre}_{e_0 e_i}(e_i)$. Suppose that each edge e_i $(0 \leq i \leq k)$ has local coordinates ranging from 0 to d_i. We dispose sequentially all the edges of R in the positive real line, starting for example at position p, *i.e.*, $e_i = (l_i, u_i]$ with $l_0 = p$, $u_0 = p + d_0$ and for all $1 \leq i \leq k$, $l_i = u_{i-1}$ and $u_i = l_i + d_i$. Hence, a point on edge e_i with local coordinates λ will be situated on the real line $\mathbb{R}$ in position $l_i + \lambda$ (see Fig. 4). We proceed in the same way for the other regions of PCD_i.

Let $\mathsf{Succ}_{e_i e_j}(\lambda) = a_i\lambda + b_i$ be a one-step successor, we define a function f as follows:

$$f(z) = A_i z + B_i \text{ if } z \in I_i$$

where $A_i = a_i$ and $B_i = b_i + l_j - a_i l_i$

We show now that for $\lambda_0 \in e_i$ and $\lambda_f \in e_j$, $\mathsf{Succ}_{e_i e_j}(\lambda_0) = \lambda_f$ iff $z_f = f(z_0)$. Let $\mathsf{Succ}_{e_i e_j}(\lambda_0) = \lambda_f = a_i\lambda_0 + b_i$ such that λ_0 and λ_f have coordinates $z_0 = l_i + \lambda_0$ and $z_f = l_j + \lambda_f$ on $\mathbb{R}$. Thus

$$\begin{aligned} \lambda_f = a_i\lambda_0 + b_i \quad & \text{iff } z_f - l_j = a_i(z_0 - l_i) + b_i \\ & \text{iff } z_f = a_i z_0 + (b_i + l_j - a_i l_i) \\ & \text{iff } z_f = A_i z_0 + B_i \\ & \text{iff } z_f = f(z_0). \end{aligned}$$

We have then constructed a function f for each one-step successor. The PAM $\mathcal{A}$ corresponding to the PCD of location ℓ_i is defined then as the function that consists of the body of all the functions f above. Up to now we have encoded just a simple PCD, it remains to encode the jumps from location ℓ_i to location ℓ_j in order to simulate a HPCD by a PAM. This is done in the same way as before, since the reset are edge-to-edge affine functions.

From the above results we have that $\text{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ iff $\text{Reach}(\mathcal{A}, z_0, z_f)$. □

The above simulation is valid only if the HPCD $\mathcal{H}$ is bounded; infinite edges of an unbounded HPCD cannot be arranged on the real line.

In order to prove that HPCDs simulate PAMs we should: (1) Encode an initial and final point of $\mathcal{A}$ by points on some edges on $\mathcal{H}$; (2) Represent a configuration of $\mathcal{A}$ by a configuration of $\mathcal{H}$; (3) Simulate one-step computation of $\mathcal{A}$ by some trajectory segment (many-steps successor) on $\mathcal{H}$.

Lemma 3.4 (HPCDs simulate PAMs). *Every PAM $\mathcal{A}$ can be simulated by a HPCD. Every bounded PAM $\mathcal{A}$ can be simulated by a bounded HPCD.*

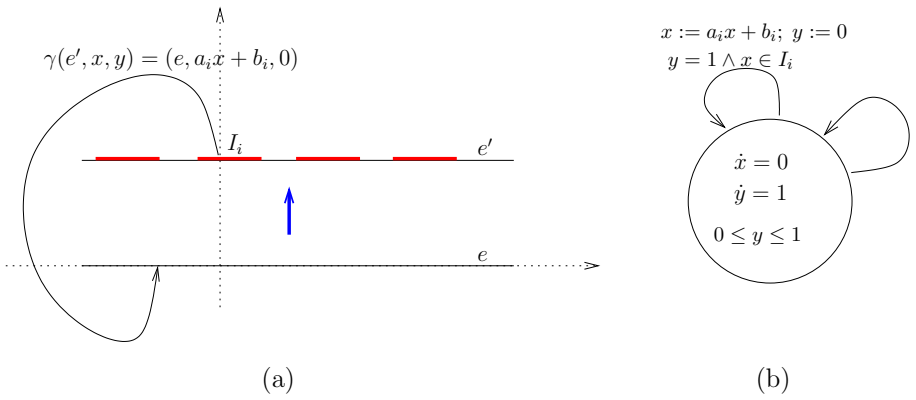


Figure 5: (a) The HPCD that simulates a PAM; (b) An equivalent SA.

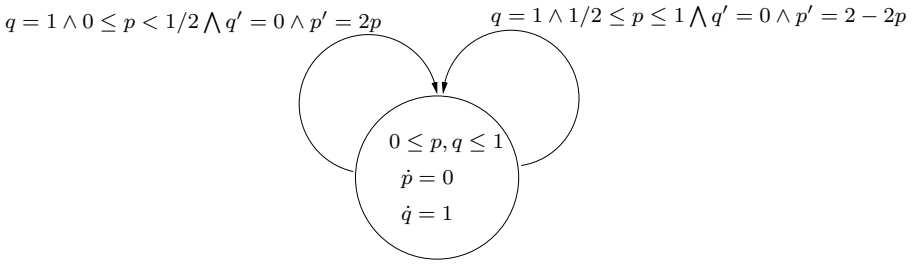


Figure 6: One-State Tent Map HPCD.

Proof: Let $\mathcal{A}$ be defined by $f(z) = a_i z + b_i$ if $z \in I_i$ for $i \in \{1, \cdots, k\}$, where I_i are rational intervals. We define a one-location HPCD with a one-region PCD defined by $y \geq 0 \wedge y \leq 1$, *i.e.*, there are two edges $e \equiv y = 0$ and $e' \equiv y = 1$, and dynamics defined by vector $(0, 1)$ as shown in Fig. 5-(a). There are as many transitions as intervals I_i of the PAM. The guards are of the form $y = 1 \wedge x \in I_i$, with their reset functions being of the form $\gamma(x, y) = (a_i x + b_i, 0)$. The initial point z_0 of the PAM is encoded as a point $(x_0, y_0) \in e$ with abscissa $x_0 = z_0$. Hence, it is easy to see that points with abscissae $f^n(z_0)$ will be visited in sequence, establishing simulation.

If moreover $\mathcal{A}$ is bounded, the stripe region $0 \leq y \leq 1$ can be replaced by the rectangular one: $0 \leq y \leq 1 \wedge -M \leq x \leq M$ with M large enough, and this yields the required bounded HPCD. □

From the above two lemmas, we have then the following theorem.

Theorem 3.5 (bounded HPCDs are equivalent to bounded PAMs). *When restricted to bounded systems,* REACH_{HPCD} *is decidable iff* REACH_{PAM} *is.* □

Example 3.1. *Consider the PAM describing the Tent Map [29]:*

$$f(x) = \begin{cases} 2x + 0, & \text{if } x \in [0, 1/2) \quad (\equiv I_1) \\ -2x + 2, & \text{if } x \in [1/2, 1] \quad (\equiv I_2) \end{cases}$$

The HPCD simulating this PAM is shown in Fig. 6. □

3.2. PCDs with Translational Resets

Here we show that a bounded PAM can be simulated by a PCD augmented with translational resets of the form $\mathbf{x}' = \mathbf{x} + \mathbf{c}$. This result is achieved by realizing the affine transformations by intersecting the rectilinear flow with orthogonal edges and by simulating the PAM variable (x) on the two PCD variables (p and q) in turns. The next iterate is computed based on the variable that carries either a copy of the current value or of the previous iterate. Using the boundedness of the PAM, we can see that the PCD variables lie in a bounded region in every state. Thus, by separating the states (along each dimension) by a distance greater than the absolute maximum value of the variables, we can ensure that state invariants are non-overlapping. The following evident lemma simplifies the proof.

Lemma 3.6. *Every bounded PAM is equivalent to a 1-dim "positive" PAM where all intervals are positive.*

Proof: Just shift the PAM to the positive semi-axis. □

Results proved over these positive PAMs are thus applicable to general bounded PAMs as well. We can now prove a key result:

Theorem 3.7. *A bounded PAM can be simulated by a PCD with translational resets.*

Proof: Consider an equivalent positive PAM $f(x) = a_i x + b_i \ , \ x \in I_i$ $(i = 1, 2, \cdots, n)$, with the intervals enumerated in the increasing order. Let L be a number such that $\forall i, \ L > b_i$, and L is greater then the right extremity of the rightmost interval I_n. Corresponding to the i-th function of the PAM, we will have two states P_i and Q_i. In P_i, the variable p flows from $p_0 = b_i$ to $x_{n+1} (\equiv b_i + a_i x_n)$ at the rate $\dot{p} = a_i$. The other variable q drops from $q_0 = x_n$ to 0 at the rate $\dot{q} = -1$. The guard $q = 0$ thus ensures that the system spends $t = q_0$ time in this state. This allows the affine term $a_i x_n$ to be computed, without using comparative guards or affine resets. In the Q states, the roles of p and q are reversed, *i.e.*, q uses p's value to grow to the next iterate, while p just drops to 0, effectively keeping track of time. From P_i, there are transitions to each possible state Q_j. The variable p retains the value it just computed, while q is reset to the constant portion (b_j) of the next iterate of x. In Q_j, q will accumulate the rest of its target value ($a_j x$) by flowing for time x (stored in p) at the rate a_j. Similarly, from Q_i, there are transitions to each possible state P_j, while there are no transitions within P-states or within Q-states.

The above expressions are adjusted, now assuming that each state is associated with a different large constant "base", *i.e.*, x becomes $L_{S_i} + x$ in state

S_i, where L_{S_i} is the base value derived from the value L defined above. As x evolves in a state, p and q will not be able to inadvertently cross over to another state because of the designed difference in their base values; the need to adjust the base value during state transitions leads to translational resets. The full details of the construction are provided below:

- Corresponding to the i-th function of the PAM, we have two states P_i and Q_i associated with the constants $L_{P_i} = 4iL - 3L$ and $L_{Q_i} = 4iL - L$.
- In P_i, p grows at rate $\dot{p} = a_i$ from $L_{P_i} + p_0 (= b_i)$ to $a_i q_0 (= x_n) + b_i + L_{P_i}$, while q drops from $q_0 + L_{P_i}$ to L_{P_i} at the rate $\dot{q} = -1$. Here q_0 denotes the unscaled previous iterate x_n, using which x_{n+1} is being computed by spending exactly $t = q_0$ time in this state.
- Q_i behaves exactly as above with p and q swapped, *i.e.*, this corresponds to the case where q grows to the next iterate, while p just drops to L_{Q_i}.
- In P_i and Q_i, the values of p and q are both bounded by $\{(L_{P_i/Q_i} - L, L_{P_i/Q_i} + L)\}$, which is equal to $\{(4iL - 4L, 4iL - 2L)\}$ in P_i and $\{(4iL - 2L, 4iL)\}$ in Q_i. Clearly, none of rectangular regions can overlap.
- From P_i, there are transitions to each possible state Q_j with guard $q = L_{P_i} \wedge p \in I_j$, *i.e.*, "$p$ has reached the next iterate of x" *and* "p is in the interval corresponding to the j-th PAM function". The reset (translational) is $p' = p - L_{P_i} + L_{Q_j} \wedge q' = q - L_{P_i} + L_{Q_j} + b_j$, *i.e.*, "$p$, which holds the current value of x, is translated to the range of the destination state (to prevent overlap)" *and* "q is in fact reset to the constant portion (b_j) of the next iterate of x" (since the value of q before the transition was a constant, this can be expressed by a translation). The portion proportional to x_n (*i.e.*, $a_j x_n$) will be gained by flowing for time x_n (stored in p) with slope a_j.
- Similarly, from Q_i, there are transitions to each possible state P_j. There are no transitions within P-states or within Q-states.

This PCD with translational resets simulates the PAM, as p and q take turns simulating x. It can be seen that x_f is reachable from x_0: (i) if $(p = L_{Q_j} + x_f, q = L_{Q_j} + b_j)$; or (ii) if $(p = L_{Q_j} + b_j, q = L_{Q_j} + x_f)$ is reachable from the starting state $(p = x_0 + L_{Q_k}, q = L_{Q_k} + b_k)$. Here k, j are indices of intervals containing x_0 and x_f, that is $x_0 \in I_k$, $x_f \in I_j$. The disjunction is necessary because p reaches only even iterates and q reaches only odd iterates of x_0. □

Example 3.2. *The Tent Map can be simulated by a PCD with translational resets, using two variables p and q and $2 \times 2 = 4$ states. Setting $L = 3 (> max(r_n, b_i) = 2)$, we get $L_{P_1} = 3, L_{Q_1} = 9, L_{P_2} = 15, L_{Q_2} = 21$. Thus:*

> *P_1: flows $\dot{p} = a_1 = 2$ and $\dot{q} = -1$, with transitions:*
> *$\rightarrow$ Q_1: guard $q = 3 \wedge p \in [3 + 0, 3 + 1/2)$, reset $p' = p - 3 + 9 = p + 6 \wedge q' = q - 3 + 9 + 0 = q + 6$*

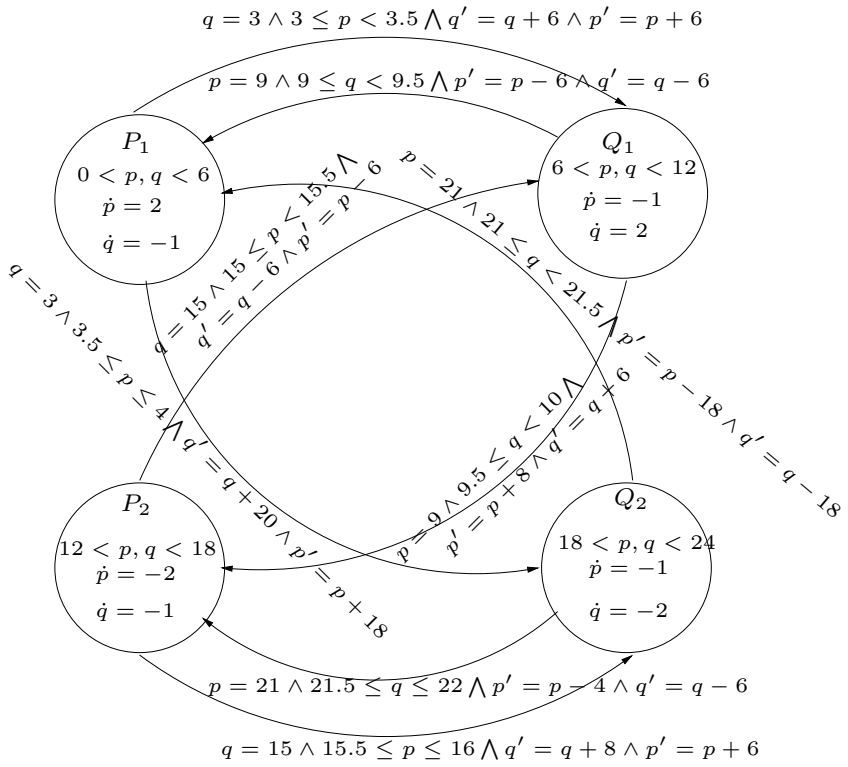


Figure 7: PCD with Translational Resets simulating the Tent Map.

$\rightarrow Q_2$: *guard* $q = 3 \wedge p \in [3 + 1/2, 3 + 1]$, *reset* $p' = p - 3 + 21 = p + 18 \wedge q' = q - 3 + 21 + 2 = q + 20$

P_2: *flows* $\dot{p} = a_2 = -2$ *and* $\dot{q} = -1$, *with transitions:*

$\rightarrow Q_1$: *guard* $q = 15 \wedge p \in [15 + 0, 15 + 1/2)$, *reset* $p' = p - 15 + 9 = p - 6 \wedge q' = q - 15 + 9 + 0 = q - 6$

$\rightarrow Q_2$: *guard* $q = 15 \wedge p \in [15 + 1/2, 15 + 1]$, *reset* $p' = p - 15 + 21 = p + 6 \wedge q' = q - 15 + 21 + 2 = q + 8$

Q_1: *flows* $\dot{q} = a_1 = 2$ *and* $\dot{p} = -1$, *with transitions:*

$\rightarrow P_1$: *guard* $p = 9 \wedge q \in [9 + 0, 9 + 1/2)$, *reset* $q' = q - 9 + 3 = q - 6 \wedge p' = p - 9 + 3 + 0 = p - 6$

$\rightarrow P_2$: *guard* $p = 9 \wedge q \in [9 + 1/2, 9 + 1]$, *reset* $q' = q - 9 + 15 = q + 6 \wedge p' = p - 9 + 15 + 2 = p + 8$

Q_2: *flows* $\dot{q} = a_2 = -2$ *and* $\dot{p} = -1$, *with transitions:*

$\rightarrow P_1$: *guard* $p = 21 \wedge q \in [21 + 0, 21 + 1/2)$, *reset* $q' = q - 21 + 3 = q - 18 \wedge p' = p - 21 + 3 + 0 = p - 18$

$\rightarrow P_2$: *guard* $p = 21 \wedge q \in [21 + 1/2, 21 + 1]$, *reset* $q' = q - 21 + 15 = q - 6 \wedge p' = p - 21 + 15 + 2 = p - 4$.

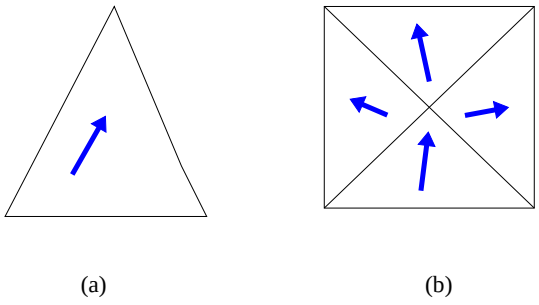


Figure 8: Two forbidden configurations in $\mathrm{PCD}_{2\mathrm{m}}$: (a) Flow vector parallel to an edge; (b) Branching in a vertex.

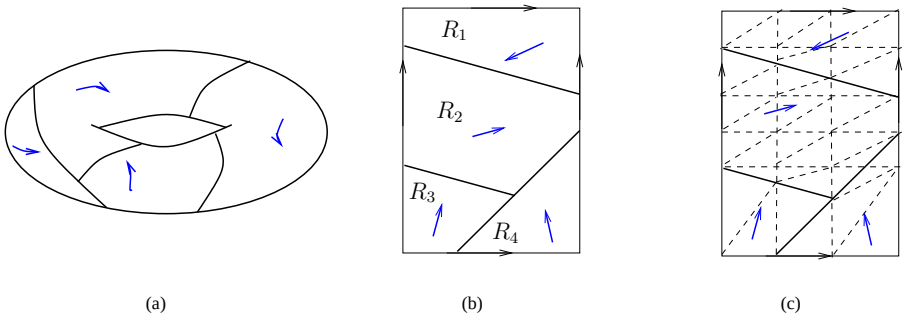


Figure 9: A $\mathrm{PCD}_{2\mathrm{m}}$ on the torus: Three views.

Clearly, p and q take turns simulating the PAM variable x. (See Fig. 7) □

3.3. PCDs on 2-Dimensional Manifolds

In this section we will study the reachability problem of PCDs defined on surfaces, or 2-dimensional manifolds (introduced in section 2.4). To define a PCD on a triangulated surface S, a PCD should be defined on each of its triangles. We call this class of systems *PCD on 2-dimensional manifolds* ($\mathrm{PCD}_{2\mathrm{m}}$). We impose a couple of restrictions on the dynamics (see Fig. 8):

- The flow vector is never parallel to an edge of its triangle.
- Every vertex can be an input vertex for at most one triangle.

The first assumption is just for simplicity and can be removed. The second one ensures unicity of trajectories and excludes "branching" in vertices. In Fig. 9 we define a PCD on a torus and show how to represent it as a family of PCDs on triangles. As before, a point $\mathbf{x}_f$ is *reachable* from another point $\mathbf{x}_0$ if there exists a trajectory from $\mathbf{x}_0$ to $\mathbf{x}_f$, defining the reachability problem:

Problem 3.8. *Given a* $\mathrm{PCD}_{2\mathrm{m}}$ $\mathcal{H}$*, is point* $\mathbf{x}_f$ *reachable from* $\mathbf{x}_0$*?*

We show that indeed the decidability of Problem 3.8 is an open problem, showing as before that $\mathsf{REACH}_{\mathrm{PCD}_{2\mathrm{m}}}$ is equivalent to REACH_{PAM} for a large subclass of PAMs. We recall that $V(f) \subset \mathbb{R}$ is the finite set of all the "vertices" (*i.e.*, extremities of intervals) of a PAM f.

The idea of translation from a PCD on a manifold to a PAM is as follows. By definition a manifold is a finite collection of triangles with identified edges, and a PCD defines a constant flow in each triangle. This flow defines a piecewise-affine successor relation between edges of the triangle. Identification between edges gives another piecewise affine relation between edges (only identification between an output edge and an input one matters for reachability). By taking the union of the successor relations and the useful identifications we can obtain a PAM, defined on the edges of the triangulation of the manifold, which simulates the PCD. If we want to put it onto the real line, we are obliged to"tear" some edges from each other in certain vertices and dispose all the edges on the line. Formally speaking, this operation can destroy continuity or injectivity. Nonetheless the PAM remains continuous and injective if we identify points on the line corresponding to the same vertex. We call such PAMs regular (see Definition 3.9 below).

Conversely, given a regular PAM, we first put it on a horizontal stripe with vertical flow (like in Lemma 3.4). Next we replace every affine map from a closed interval to another one by identification of the source and the target intervals on the upper and lower edges of the stripe (injectivity and continuity insure that this identification is correctly defined). In virtue of manifold classification theorem (Theorem 2.4) we obtain a piecewise constant flow on a manifold with some disks removed. "Sewing" these holes by disks with trivial dynamics we obtain a required PCD on a manifold.

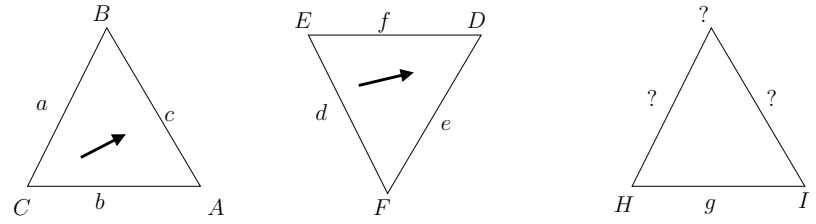
In what follows we sketch these constructions with more details.

Definition 3.9. *A PAM* f *defined on a compact domain* $D(f)$ *is* regular *if there exists an equivalence relation* $\sim$ *on the set* $V(f)$ *of its vertices such that* $f/\sim$ *is an injective continuous function.* □

In other words, the function f should become injective and continuous after identifying finitely many groups of finitely many points. It should thus satisfy the following:

- f is injective on the interior of its intervals $D(f) \setminus V(f)$ (we say that f is almost injective);
- for any two vertices u and v, we have that $f(u) \sim f(v)$ if and only if $u \sim v$;
- values of f in all the vertices coincide (up to $\sim$) with left and right limits of adjacent intervals. Formally, whenever $f(x) = ax + b$ for $x \in (u, v)$, it should be that $f(u) \sim au + b$ and $f(v) \sim av + b$.

We denote by $\mathrm{PAM}_{\mathrm{reg}}$ the class of regular PAMs, and the corresponding reachability problem by $\mathsf{REACH}_{\mathrm{PAM}_{\mathrm{reg}}}$.

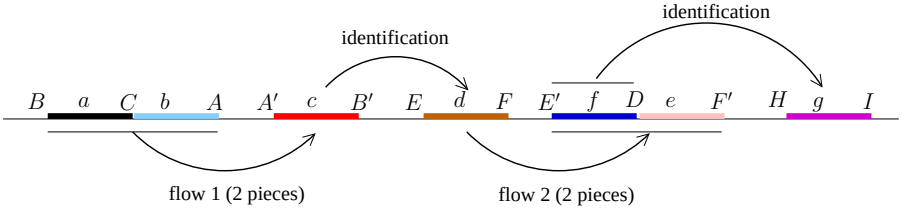


Two input edges, one output

Two output edges, one input

c identified with d
f identified with g

(a)



(b)

$$
\begin{array}{l}
A \sim A' \sim F \\
B \sim B' \sim E \sim H \\
D \sim I \\
F \sim F'
\end{array}
$$

(c)

Figure 10: From a $\mathrm{PCD}_{2\mathrm{m}}$ to a regular PAM: (a) A fragment of a $\mathrm{PCD}_{2\mathrm{m}}$ $\mathcal{H}$; (b) The $\mathrm{PAM}_{\mathrm{reg}}$ F simulating $\mathcal{H}$; (c) The equivalence relation $\sim$.

Lemma 3.10 ($\mathrm{PAM}_{\mathrm{reg}}$ simulate $\mathrm{PCD}_{2\mathrm{m}}$). *Every* $\mathrm{PCD}_{2\mathrm{m}}$ *can be simulated by a regular PAM.*

Sketch of the proof: Let $\mathcal{H}$ be a $\mathrm{PCD}_{2\mathrm{m}}$. The reduction is analog to the simulation of HPCDs by PAMs (see Lemma 3.3), but special effort is necessary to ensure regularity. We suppose that the support manifold is triangulated, and we consider all the triangles separately. We will represent every edge e of every triangle by a line segment $r(e)$ of the same length as e on $\mathbb{R}$ (and every point $x \in e$ is represented by a point $r(x) \in r(e)$). These segments $r(e)$ are positioned in groups of one or two. Namely when a triangle has two input edges their representations should be adjacent on $\mathbb{R}$, and similarly for two output edges (see Fig. 10-(a)). In all the other cases the representing intervals on $\mathbb{R}$ should be put apart of each other. All these representing intervals constitute the support of the PAM f. The mapping itself is defined as a finite union of affine and piecewise affine mappings of two following types.

Flow-mappings For each triangle the successor map (corresponding to the flow) from its input to its output edges is in fact piecewise affine (with two pieces). We reproduce this map on the representing intervals. Thus representing intervals of its input edges are mapped onto representatives of its output edges – it can be done by a two-piece map (see example on Fig. 10).

Identification-mappings For each pair of identified edges e_1 and e_2, if one of them (say e_1) is input, and the other one (e_2) is output, then the representative $r(e_2)$ of the output edge is mapped onto the representative $r(e_1)$ of the input one. (If two input or two output edges are identified, then the flow of the $\mathrm{PCD}_{2\mathrm{m}}$ cannot traverse this edge, and it is useless to map their representatives to each other.)

The PAM obtained is not injective nor continuous, but it becomes so if we glue together all its vertices corresponding to identified vertices of $\mathcal{H}$ (this is a finite equivalence relation $\sim$). Hence, the PAM is regular.

The relation between $\mathcal{H}$ and f can be described as follows: For any trajectory $\xi(t)$ of $\mathcal{H}$ take the points x_i where it enters and exits every triangle. These points belong to the edges of the triangulations, and hence are represented by some points $r(x_i)$ in the PAM f. The sequence $\{r(x_i)\}_i$ is in this case a trajectory of the PAM f.

The converse is also true, every trajectory $\{y_i\}_i$ of f represents in this way some trajectory of $\mathcal{H}$. Indeed, we should just take the sequence $\{x_i = r^{-1}(y_i)\}$ in $\mathbb{R}^2$ and build a trajectory of $\mathcal{H}$ that passes through the points x_i. Whenever the mapping from y_i to y_{i+1} is a flow-mapping, we connect x_i with x_{i+1} by a straight line segment. By construction of f, this segment is always parallel to the flow vector of $\mathcal{H}$ and can hence be seen as a trajectory segment of $\mathcal{H}$. By the same construction, whenever the mapping from y_i to y_{i+1} is an identification-mapping in f, then the points x_i and x_{i+1} are identified and represent the same point in $\mathcal{H}$. Hence, we have obtained a trajectory of $\mathcal{H}$ which hits the edges in the points $\{x_i\}$. □

The example depicted in Fig. 10 helps understanding the construction. The part (a) of the figure presents three triangles of a $\mathrm{PCD}_{2\mathrm{m}}$ with corresponding flow vectors (for the third triangle we only consider its input edge). Notice that AB is identified with EF and ED with HA. The part (b) presents the PAM simulating the $\mathrm{PCD}_{2\mathrm{m}}$. We have cut the perimeter of the first triangle in the vertices B and A in order to put the three edges on the real line. Similarly for two other triangles. In this way every edge of every triangle is represented in the PAM. The flow of the he $\mathrm{PCD}_{2\mathrm{m}}$ brings every point on AC and CB to a point on AB. This is represented in the PAM by a 2-piece affine mapping from ACB onto $A'B'$ designated as "flow 1" on Fig. 10 (b). Similarly for "flow 2" from $A'B'$ to $E'DF'$. Last, but not least the identifications between edges are represented by other affine mappings in the PAM: from $A'B'$ to EF and from $E'D$ to HI. The PAM obtained has discontinuities in C and D, but after identifying vertices according to (c) it becomes injective and continuous.

We now turn our attention to the simulation of a $\mathrm{PAM}_{\mathrm{reg}}$ by a $\mathrm{PCD}_{2\mathrm{m}}$.

Lemma 3.11 ($\mathrm{PCD}_{2\mathrm{m}}$ simulate $\mathrm{PAM}_{\mathrm{reg}}$). *Every regular PAM can be simulated by a* $\mathrm{PCD}_{2\mathrm{m}}$.

Sketch of the proof: Let $\mathcal{A}$ be a regular PAM defined as $f(z) = f_i(z) = a_i z + b_i$ if $z \in I_i$ for $1 \leq i \leq n$. We obtain a $\mathrm{PCD}_{2\mathrm{m}}$ by a construction similar to Lemma 3.4. In a polygon $R \supset [-M; M] \times [0; 1]$ (with M large enough), the dynamics is defined by vector $(0, 1)$[6]. To realize the function f by identification of edges, we partition the edges of the PAM (see Fig. 11: on the bottom of the rectangle R we define $I_i^k = (f_i(I_i) \cap I_k) \times \{0\}$, on the top, we define $J_i^k = (I_i \cap f_i^{-1}(I_k)) \times \{1\})$. Almost injectivity of the PAM $\mathcal{A}$ guarantees that these intervals do not overlap. Continuity guarantees that a whole closed edge is identified with another edge (up to $\sim$-equivalence on vertices).

Next we identify each non-empty J_i^k with I_i^k via the function f_i (which is an affine bijection between these two edges). It is easy to find a triangulation such that I_i^k and J_i^k are its edges, hence we have represented our system as a PCD on a compact surface with boundary.

By the Classification Theorem for Surfaces with Boundary (see Theorem 2.4) we have that this surface is equivalent to a manifold with some disks removed and we obtain then a $\mathrm{PCD}_{2\mathrm{m}}$ just "sewing" the disks. We associate with these disks a zero slope vector. □

From the above two lemmas the Turing equivalence of $\mathsf{REACH}_{\mathrm{PAM}_{\mathrm{reg}}}$ and $\mathsf{REACH}_{\mathrm{PCD}_{2\mathrm{m}}}$ restricted to points on edges is immediate. On the other hand notice that $\mathsf{REACH}_{\mathrm{PCD}_{2\mathrm{m}}}$ restricted to points on edges is equivalent to $\mathsf{REACH}_{\mathrm{PCD}_{2\mathrm{m}}}$. Indeed, in order to check whether $\mathbf{x}_f$ is reachable from $\mathbf{x}_0$, it suffices to draw a trajectory from $\mathbf{x}_0$ till it hits some edge in a point $\mathbf{x}'_0$, to draw a trajectory backwards from $\mathbf{x}_f$ till it hits some edge in a point $\mathbf{x}'_f$, and check reachability from $\mathbf{x}'_0$ to $\mathbf{x}'_f$ (both points belong to edges).

[6] In order to respect the definition of regular $\mathrm{PCD}_{2\mathrm{m}}$ we take a polygon without vertical edges rather than a rectangle.

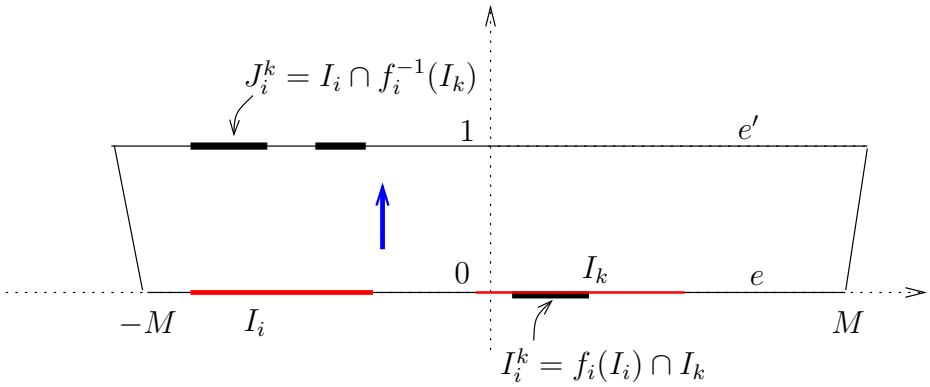


Figure 11: Simulation of a $\mathrm{PCD}_{2\mathrm{m}}$ by a $\mathrm{PAM}_{\mathrm{reg}}$: Edge J_i^k identified with I_i^k via f_i.

Theorem 3.12 ($\mathrm{PCD}_{2\mathrm{m}}$ are equivalent to $\mathrm{PAM}_{\mathrm{reg}}$). *Reachability for* $\mathrm{PCD}_{2\mathrm{m}}$ *is decidable iff reachability for regular PAMs is.* □

3.4. Other Open Subclasses

Various other intermediate subclasses of low dimensional hybrid automata simulate a PAM. We now present some of the interesting cases, based on earlier proofs and constructions.

3.4.1. Variants of Linear, Timed and Rectangular Automata

We first establish some simple corollaries of Theorem 3.5. Observing the construction in the proof of the theorem, we see that the only features of HPCDs / LHAs used can be captured by the simple LHA class (see Fig. 5-(b) for the simple LHA corresponding to the HPCD in Fig. 5-(a)).

Definition 3.13. *A* simple *LHA (SA) is a 2-dimensional LHA with only one location ℓ and 2 variables: one clock y, and one memory cell x. The invariant is of the form $C \leq y \leq D$, guards are of the form $y = D \wedge x \in I$, and resets of the form $\gamma(x, y) = (ax + b, c)$.* □

The next result follows immediately from the proof of Theorem 3.5.

Corollary 3.14 (SAs are equivalent to PAMs). *Reachability for SAs is decidable iff reachability for PAMs is.* □

We deduce that the same holds for any class between SAs and HPCDs.

Corollary 3.15 (Intermediate classes are equivalent to PAMs). *For any class $\mathcal{C}$ of hybrid automata, such that $SA \subseteq \mathcal{C} \subseteq HPCD$, reachability for $\mathcal{C}$ is decidable iff reachability for PAMs is.* □

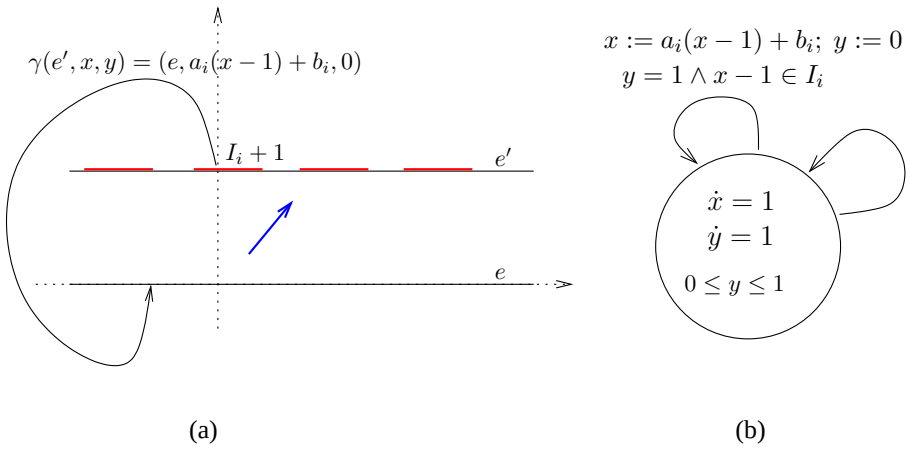


Figure 12: (a) Another HPCD that simulates a PAM; (b) The corresponding AUTA.

Another simple corollary concerns a variant of updatable timed automata [30, 14] with affine resets, based on the proof of Theorem 3.5.

Definition 3.16. *A 2-clocks* affine updatable timed automaton *(2AUTA) is a LHA with two clocks x and y, invariants of the form $C \leq y \leq D$, guards of the form $y = D \wedge x \in I$ and resets of the form $\gamma(x, y) = (ax + b, 0)$.* □

Corollary 3.17 (AUTAs are equivalent to PAMs). *Reachability for 2AUTAs is decidable iff reachability for PAMs is.*

Sketch of the proof: In Lemma 3.4 a HPCD $\mathcal{H}$ (see Fig. 5) that simulates a PAM was built. We obtain another HPCD $\mathcal{H}'$ applying an affine transformation to $\mathcal{H}$, where the edge e remains unchanged whereas e' is translated by one unit to the right. $\mathcal{H}'$ is represented in Fig. 12-(a), where given $I = [l, u]$ the notation $I + 1$ is a short for $[l + 1, u + 1]$. It is not difficult to see that the automaton of Fig. 12-(b) is a 2AUTA equivalent to $\mathcal{H}'$. □

Next, we extend the constructions devised for proving Theorem 3.7 to characterize other extensions of timed automata that can simulate a PAM. In Corollary 3.17 we have shown that the affine reset suffices. Our constructions below shows that it is possible to capture a PAM in a HPCD with comparative guards and simple translational resets.

Proposition 3.18. *A bounded PAM can be simulated by a 2-clock automaton, when translational resets and comparative guards are allowed.*

Proof: The following HPCD with $2n$ states of the form P_j and Q_j simulates the equivalent positive PAM. We associate a number L_{P_i/Q_i} with each discrete state that is separated from every such number by at least L in the positive and negative directions. The state P_j (and Q_j) is defined as:

- p flows from $L_{P_j} + p_0$ to $L_{P_j} + p_0 + a_j p_0 + b_j$ with $\dot{p} = +1$;
- q flows from $L_{P_j} + 0$ to $L_{P_j} + a_j p_0 + b_j$ with $\dot{q} = +1$;
- The discrete transitions will be of the form $P_j \to Q_k$ with guard $a_j p - (1 + a_j)q + b_j + L_{P_j} = 0 \wedge q \in I_k$ and reset $p' = 0 + L_{Q_k} \wedge q' = q - L_{P_j} + L_{Q_k}$.

□

These ideas can be adapted to show that decidable initialized rectangular automata [13, 12] can simulate a PAM, when extended with comparative guards or when uninitialized. We only prove the latter here.

Proposition 3.19. *A bounded PAM can be simulated by a 2-dim (uninitialized) rectangular automaton.*

Proof: The following rectangular automaton with $2n$ states of the form P_j and Q_j simulates the equivalent positive PAM. The state P_j is defined as follows, with the other states defined symmetrically:

- While entering this discrete state, p has the current value of x;
- p flows from p_{in} to 0 with $\dot{p} = -1$;
- q flows from b_j to $b_j + p_{in} a_j$ with $\dot{q} = a_j$;
- The invariant is $p > 0 \wedge q \lesseqgtr b_j$, with the inequality determined by a_j's sign;
- The discrete state transitions are of the form $P_j \to Q_k$ with guard $p = 0 \wedge q > 0$ and reset $p' = b_k \wedge q' = q$. Since q is not reset even though its flow changes in the next state, this HA is not an "initialized" automaton.

□

3.4.2. Variants of HPCDs

The proof methodology of Theorem 3.7 can be extended to characterize the simplest HPCD subclass that can simulate a PAM without using resets at all (*i.e.*, all the resets are identity relations); our construction uses both overlapping invariants and comparative guards.

Proposition 3.20. *A bounded PAM can be simulated by a HPCD with comparative guards, 3 different flows* $+1, -1, 0$ *for each variable and no resets.*

Proof: Consider a PAM $f(x) = a_i x + b_i \ , \ x \in I_i \ , \ i = 1, 2, \cdots, n$. Once again, we use the "taking-turns" idea. Unlike the proof of Theorem 3.7, we cannot initialize a variable's value at b_i as we do not have resets. So, we now have p evolving from x_{n-1} to x_{n+1}, while q remains stationary at x_n. The guard condition $p = a_i q + b_i$ makes the HA jump to the next state at the correct time. Since x_{n+1} could be greater or less than x_{n-1}, the flow will need to be $+1$ or -1 respectively. Hence, each P (and Q) state now corresponds to two states: P^+ and P^-. We will construct a HPCD with $4n$ states of the form $P_j^{\pm}$ and $Q_j^{\pm}$ that simulates this PAM; p and q will again take turns simulating x *i.e.*, $p_{2m} = p_{2m+1} = x_{2m}$ and $q_{2m-1} = q_{2m} = x_{2m-1}$, at the end of each discrete transition. Consider a state $P_j^{\pm}$ defined as follows:

- Since the initial value of x is $p_{in} \in I_j$, the variable q should flow from some q_{in} to the new value of x, that is $q_{out} = a_j p_{in} + b_j$.
- q's flow is $\dot{q} = +1$ since $q_{out} > q_{in}$ in P_j^+. To ensure this inequality, the condition $q < a_j p + b_j$ should be incorporated in all the guards leading to P_j^+. Symmetrically, in P_j^-, the opposite inequality $q'_{out} \leq q_{in}$ and flow $\dot{q} = -1$ should be used. This is also insured by the guard.
- p remains stationary (and equal to p_{in}), *i.e.*, $\dot{p} = 0$ in this state, in order to ensure that q flows to the correct amount.
- The guard condition is satisfied when q reaches q_{out}, *i.e.*, $q = a_j p + b_j$.
- The transitions out of this state are of the form $P_j^{\pm} \to Q_k^{\pm}$ only. In the next state, q stays fixed at this computed value, while p flows to the next iterant of x. In particular, the guard for jumping to Q_k^+ will be $q = a_j p + b_j \wedge q \in I_k \wedge p < a_k q + b_k$. (The last constraint will be $p \geq$ if we are jumping to Q_k^-).

The $Q_j^{\pm}$ states are defined exactly as above, with p and q interchanged. The above HPCD without resets simulates the given PAM. In particular, the reachability query "*Is x_f reachable from x_0*" is true *iff* $(Q_n^{\pm}, p = x_f, q = x_{f-1})$ *or* $(P_n^{\pm}, p = x_{f-1}, q = x_f)$ *is reachable from* $(Q_m^{\pm}, p = x_0, q = x_1)$, where x_{f-1} is some pre-image of x_f. Here indices m and n are such that $x_1 \in I_m$ and $x_{f-1} \in I_n$, and the $\pm$ signs depend on whether $x_2 > x_1$ or not and whether $x_f > x_{f-1}$ or not. There can be at most n pre-images of the target x_f, and hence n reachability queries; further, a factor of two arises from having to accomodate p or q reaching x_f (odd or even iterations). □

The idea can be further adapted to show that PAMs can also be simulated by HPCDs when augmented with origin-dependent rates (introduced in [31]) and overlapping state-invariants (HPCD_x).

Proposition 3.21. *A PAM can be simulated by an origin-dependent HPCD with rectangular guards / invariants, and identity resets.*

Proof: The following origin-dependent HPCD with $2n$ states of the form P_j and Q_j simulates the equivalent positive PAM. The state P_j is defined as follows, with other states defined symmetrically:

- p flows from p_0 to 0 with $\dot{p} = -1$;
- q flows from 0 to $a_j p_0 + b_j$ with $\dot{q} = a_j + b_j/p_0$;
- Discrete state transitions are of the form $P_j \to Q_k$ with guard being $p = 0 \wedge q \in I_k \wedge q > 0$ and identity resets. □

4. Undecidability Results

True to its "open" nature, the HPCD class does not present any direct mechanism to simulate a Turing machine (TM) or a Minsky machine (MM). Using one HPCD variable for each MM counter makes measuring one time unit difficult; storing at least one of the counters in both the variables (or both

counters in both variables) corresponds almost directly to the original problem of simulating a TM by a PAM. In this section, we show how to extend HPCDs in order to be able to simulate a Minsky Machine.

4.1. HPCDs with One Counter

Consider the class of HPCD_{1c} which are HPCDs augmented with a counter c. In each location ℓ, the state vector (x, y) evolves according to a PCD, while c remains constant. Guards have the form $P(x, y) \wedge Q(c)$ where $P(x, y)$ is as for HPCDs and $Q(c) \equiv c = 0 \mid c > 0 \mid \mathsf{true}$. Resets are as for HPCDs, but they can also increment or decrement c. We prove that the reachability problem for HPCD_{1c} is undecidable showing that an HPCD_{1c} $\mathcal{H}$ can simulate Minsky machines for which reachability is known to be undecidable.

In this section we need a concrete syntax to describe Minsky machines. In accordance with the Definition 2.8 we will represent the MM by a list of numbered instruction of the forms:

$$\begin{array}{ll} q_i: & m{+}{+},\ \textbf{goto}\ q_j \\ q_i: & m{-}{-},\ \textbf{goto}\ q_j \\ q_i: & \textbf{if}\ m = 0\ \textbf{then}\ \ \textbf{goto}\ q_j\ \textbf{else goto}\ q_k, \end{array}$$

and the similar ones for n (and also **halt**).

Proposition 4.1 (HPCD_{1c} simulate MMs). *Every Minsky machine can be simulated by a HPCD with one counter, so* HPCD_{1c} *reachability is undecidable.*

Sketch of the proof: We associate with each q_i of a Minsky machine $\mathcal{M}$ a location ℓ_i of HPCD_{1c}. In order to encode a configuration of $\mathcal{M}$ which is a triple (q_i, m, n), we represent it in $\mathcal{H}$ by (ℓ_i, x, y, c) with the point $(x, y) = (2^{-m}, 0)$ representing the first counter of $\mathcal{M}$, and $c = n$ storing the second one. The PCD associated to the location ℓ_i simulates the instruction for the state q_i. To increment or decrement m we just divide or multiply x by 2, an operation than can be performed by a PCD. To test whether $m = 0$, we check whether $x > 0.75$. All the operations on n are done directly on the counter c. Fig. 13 represents PCD simulating instructions $m{+}{+}$, $m = 0?$ and $n{+}{+}$; PCDs for the three other instructions ($m{-}{-}$, $n{-}{-}$ and $n = 0?$) are similar.
Putting all those PCDs together we obtain a HPCD_{1c} which simulates $\mathcal{M}$. □

4.2. HPCDs with Other Infinite Structures

In this section, we augment HPCD with different infinite or periodic structures allowing an infinite number of regions, or a periodic origin-dependent dynamics, or merely integrity testing in the guards. In all the cases, very simple constructions allow Minsky machines to be simulated, and hence their reachability becomes undecidable. We capture the value of both counters m and n using one continuous unbounded integer variable $x = 2^m 3^n$; the second variable y is used as a temporary variable for other computations. Incrementing and decrementing the counter correspond respectively, to multiplying and dividing

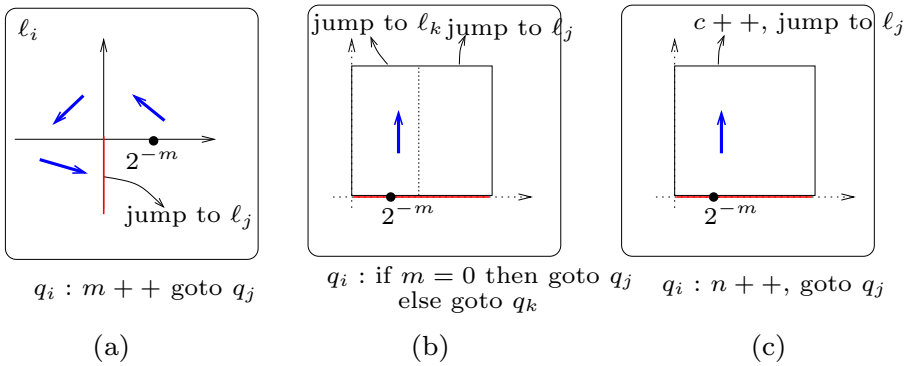


Figure 13: Sketch of the simulation of a Minsky Machine by a HPCD_{1c}: Location ℓ_i.

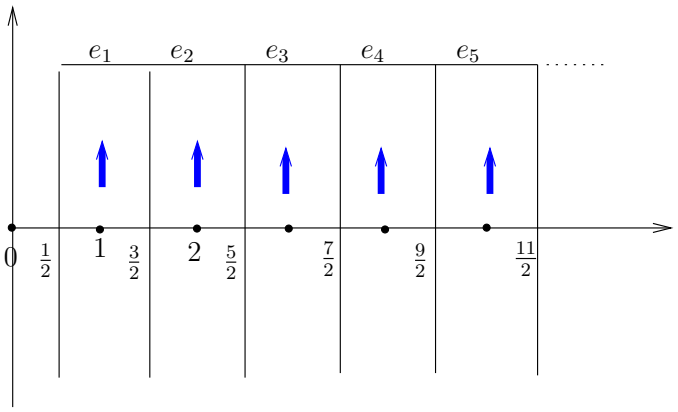


Figure 14: Sketch of the simulation of a Minsky Machine by a HPCD_∞.

by the appropriate prime factor, which can be done by a simple affine reset. The problem of simulating a Minsky Machine over an augmented HPCD now reduces to the problem of checking if $m > 0$ given the numerical value of $x = 2^m 3^n$, and being able to recover the original value of x at the end.

We will first consider HPCDs for which we relax the condition of having a finite number of regions. We call this class of systems, *HPCDs with infinite partition* (HPCD_∞). We are not going to define this class formally, since we are just interested in showing that this additional feature (having an infinite partition, even with very simple periodic structure) leads immediately to the undecidability of the reachability problem for HPCD_∞.

Proposition 4.2 (HPCD_∞ simulate MMs). *Every MM $\mathcal{M}$ can be simulated by an (unbounded) HPCD with infinite partition. Hence reachability is undecidable for* HPCD_∞.

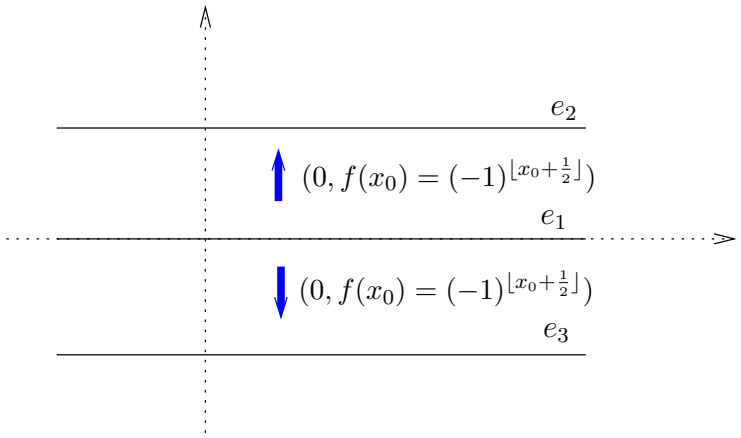


Figure 15: Simulation of a Minsky Machine by a HPCD_x, case of $m = 0$ test.

Sketch of the proof: The hybrid automaton $\mathcal{H}$ will have a location ℓ_k for each state q_k of the MM. We represent the MM counters by a point on the x-axis with the integer abscissa $x = 2^m 3^n$ in a HPCD_∞ as in Fig. 14.

If the instruction q_i increments or decrements a counter, this corresponds to an affine operation on x (division or multiplication by 2 or 3). In this case the PCD for the state ℓ_i contains one region $x > 0 \wedge 0 < y < 1$, the flow is $\dot{x} = 0, \dot{y} = 1$ (as on Fig. 14), and the jump goes to the target location and performs the necessary division or multiplication.

The only non-trivial operation is to test whether a counter is 0. Notice, that whenever x is odd we know that $m = 0$, and whenever x is not multiple of 3 we know that $n = 0$. Our periodic infinite partition allows to detect such situations. Hence, to simulate an instruction of the form q_i : **if** $m = 0$ **then** q_j **else** q_k, we make a jump from all the odd e_n edges of the location ℓ_i to the location ℓ_j, and from the even ones to ℓ_k. □

Another way of introducing "infinite patterns" is allowing continuous dynamics with some periodic behavior that depends on the initial points after a reset is done. We have already discussed *origin-dependent rate HPCDs* (HPCD_x) in Section 3.4.2. In the construction of the following proposition we will use a HPCD_x with rather particular periodic rate functions.

Proposition 4.3 (HPCD_x simulate MMs). *Every MM can be simulated by an unbounded* HPCD_x *with periodic functions as dynamics. Hence the reachability is undecidable for such systems.*

Sketch of proof: We associate with each MM-state q_i a location ℓ_i, and use the same encoding of the counters in the abscissa: $x = 2^m 3^n$. Incremening / decrementing is simulated as in the proof of Proposition 4.2 above.

For the instruction q_i : **if** $m = 0$ **then** q_j **else** q_k the location ℓ_i uses origin-dependent periodic dynamics. Its PCD_i (see Fig. 15) is defined by regions:

$R_1 : 0 < y < 1$, and $R_2 : -1 < y < 0$. The dynamics in both regions is given by the vector $(0, f(x_0))$ and the last two by (0,1). Here $f(x_0) = (-1)^{\lfloor x_0+1/2 \rfloor}$. This means, in particular, that points on e_1 with odd integer abscissae go down and hit the edge e_3, while even ones go up to e_2. Jumping from e_3 to location ℓ_j, and from e_2 to ℓ_k terminates the construction. □

The above definition allows the dynamics to be defined by any function of the initial point. To simulate a Minsky Machine, we needed to use functions that have a periodic pattern to obtain an "infinite pattern", as before. Yet another application of the same idea concerns HPCDs with integrity testing in the guards.

Proposition 4.4. *Reachability is undecidable for $HPCD_{fn-int}$, an extension of HPCDs where the guard can include a function integer(x) that returns **true** if the parameter x is an integer.*

Sketch of the proof:

- A discrete state ℓ_i corresponds to the program-state q_i of $\mathcal{M}$.
- The value of the counters is captured in the variable x as $2^m 3^n$ while y is a dummy variable typically flowing from 0 to 1 in each state. Note that the rectangles corresponding to the different discrete states are bounded (because of y) but could possibly overlap.
- MM computations: incrementing / decrementing a counter can be simulated as in the proofs of two previous results. We should address here the test instruction. The idea is that $m > 0$ if and only if $x/2$ is integer (similarly, $n > 0$ iff $x/3$ is an integer). Using this observation, we simulate the instruction q_i : **if** $m = 0$ **then** q_j **else** q_k as follows:
  1. State ℓ_{i_1} has $\dot{x} = 0$ and $\dot{y} = 1$ and jumps to state ℓ_{i_2} with guard $y = 1$ and reset $x' = \frac{1}{2}x$;
  2. State ℓ_{i_2} has $\dot{x} = 0$ and $\dot{y} = 1$ and jumps to state ℓ_j with guard $y = 1 \wedge integer(x)$ and reset $x' = 2x$, and jumps to state ℓ_k with guard $y = 1 \wedge \neg integer(x)$ and reset $x' = 2x$.

  The operations on the other counter are similar.

Clearly the HPCD simulates the 2-counter MM $\mathcal{M}$. Since reachability is undecidable for the MM, it has to be undecidable for the HPCD as well. □

5. Understanding PAMs

Having refined the decidable and undecidable frontiers of the HPCD class, we subject PAMs to a similar extend-restrain analysis.

5.1. PAM's Proximity to Undecidability

We briefly mention one contrived extension of PAMs than enables MM simulation, thus making reachability undecidable over that class.

Proposition 5.1. *PAMs that can check if a given number x can be expressed as p^{-i} (the class "PAM_{pow}"), where p is a given prime number and i is an unknown positive integer, can simulate a Minsky Machine.*

Proof: The idea is to encode the two counters m and n, and the line-number (MM instruction) l in one real number as $x = l + 2^{-m}3^{-n}$. Thus when x lies in the range $I_l \equiv (l, l+1]$, the i-th instruction of the MM needs to be executed. Thus the MM instructions can be encoded as follows:

- $(q_i : m{+}{+},$ **goto** $q_j)$ corresponds to $x' = (x-i)\frac{1}{2} + j,\ x \in I_i$;
- $(q_i : m{-}{-},$ **goto** $q_j)$ corresponds to $x' = (x-i)2 + j,\ x \in I_i$;
- $(q_i :$ **if** $m == 0$ **goto** q_j **else goto** $q_k)$ corresponds to $x' = x - i + j,\ x \in I_i \wedge x - l = 3^{-n}$ and to $x' = x - i + k,\ x \in I_i \wedge x - l \neq 3^{-n}$. □

5.2. PAM's Proximity to Decidability

The simplest PAM is one where every interval maps exactly onto another interval. Thus the mapping unwinds to a cyclical application of functions, possibly preceded by some finite prefix.

Definition 5.2 (1-dim oPAM). *A* 1-dim Onto PAM *(oPAM) is a PAM where, for every interval I_i in the PAM definition, there is an interval I_j also in the definition such that $\{a_i x + b_i | x \in I_i\} = I_j$.* □

Next we prove a crucial lemma:

Lemma 5.3. *In a 1-dim oPAM with k intervals, every point has at most $2k$ unique successors.*

Proof: If interval I_i maps onto I_j, the end points (l_i, r_i) have to map onto (l_j, r_j) or to (r_j, l_j). No other mapping is possible because of our restriction that the affine post-image of I_i has to exactly and completely overlap with I_j. Hence, there are only two possible equations linking x_j with x_i:

1. Direct $(l_i \to l_j, r_i \to r_j)$: $x_j = l_j + \frac{x_i - l_i}{r_i - l_i}(r_j - l_j)$;
2. Flipped $(l_i \to r_j, r_i \to l_j)$: $x_j = l_j + \frac{r_i - x_i}{r_i - l_i}(r_j - l_j)$.

In other words, if we define $d = \frac{x_0 - l_{x_0}}{r_{x_0} - l_{x_0}}$, only the points that are $l_j + d(r_j - l_j)$ or $l_j + (1-d)(r_j - l_j)$ are ever reachable. Thus, every interval has only two possible reachable points from a given x_0. Since there are k intervals, after $2k$ iterations all possible successors would have been explored, and there will be a cycle of period $\leq 2k$ in the path. □

This observation about exactly onto affine maps leads to the following:

Theorem 5.4. *Reachability is decidable for 1-dim oPAMs.*

Proof: The reachability query can be verified after the partitioning algorithm converges, yielding several non-overlapping "interval-node-paths" of the form $P_i \to P_{i+1} \cdots \to P_j \to P_{j+1} \cdots \to P_j$ *i.e.*, a cycle of interval-nodes possibly preceded by a linear path of interval-nodes. If the given x_0 and x_f do not lie on the same interval-node-path, then x_f is unreachable from x_0. Otherwise, numerically iteration is performed from x_0. If x_f is not reached in $2k$ steps, where k is the length of the interval path containing x_0, we can conclude that x_f is unreachable using Lemma 5.3 above. Hence, x_f is reachable *iff* it is encountered on this $2k$-long path starting at x_0. □

Remark. We can quickly conclude that x_f is unreachable if $\frac{x_f - x_{l_f}}{x_{r_f} - x_{l_f}}$ is not equal to $\frac{x_0 - x_{l_0}}{x_{r_0} - x_{l_0}}$ or $\frac{x_{r_0} - x_0}{x_{r_f} - x_{l_f}}$, where $[x_{l_f}, x_{r_f}]$ and $[x_{l_0}, x_{r_0}]$ are the interval partitions containing x_f and x_0 respectively.

Example 5.1. *$f(x) = 2x + 1/3,\ x \in [0, 1/3) (\equiv I_1)$ and $f(x) = 1/2 - x/2,\ x \in [1/3, 1] (\equiv I_2)$ is an oPAM as $f([0, 1/3]) = [1/3, 1]$ and $f([1/3, 1]) = [0, 1/3]$. Points reachable from $x_0 = 1/4$ are: $x_1 = 2/4 + 1/3 = 5/6, x_2 = 1/2 - 5/12 = 1/12, x_3 = 2/12 + 1/3 = 1/2, x_4 = 1/2 - 1/4 = 1/4 = x_0$, as expected.* □

5.3. An Approximate Reachability Algorithm

Reachability is easily semi-decidable for PAMs, with the procedure being iterating $x_0, f(x_0), f(f(x_0)), \cdots$ until x_f is reached. If x_f is not reachable, this algorithm will never converge. We present a simple algorithm for approximating the reachable points (see Alg. 1), where the intervals I_i of the PAM are partitioned, until all the successors (post-images) of points in one interval map onto exactly one complete interval (an extension of this idea was presented in [32]). Whenever this algorithm terminates (yielding a 1-dim oPAM), it computes the exact reachability relation. If we interrupt it after a certain number of iterations, it computes an under-approximation of reachability.

6. Related Work

The reachability question is decidable for certain classes of hybrid systems. In [33], it was shown that reachability is decidable for timed automata (TA), which are a particular case of hybrid automata where all the variables have slope 1. In [7], the decidability of the same problem for 2-dim PCDs was proved. In [30], some extensions of TA are considered (*updatable timed automata*) for which the decidability of the emptiness problem is studied. It has been shown that the reachability problem for multirate automata (a hybrid automaton where the variables run at any constant slope; see [34, 35]) and rectangular automata [12, 13] is decidable, under certain conditions. Some more decidability results were given for subclasses of linear hybrid systems, *extended integrator graphs* [36], and for timed graphs with one stopwatch [37].

On the other hand and not surprisingly, many undecidability results have been recorded. In [12], it was shown that the reachability problem for rectangular automata with at least 5 clocks and one two-slope variable (with rational

Algorithm 1: Over-Approximation of PAM Reachability

1. Let the initial partition P be the set of PAM intervals $\{I_i\}$
2. Pick an interval P_i in P and calculate its post-image P_i'. Let P_i' span the intervals $P_l, P_{l+1}, \cdots, P_{r-1}, P_r$.
3. P_i' induces $r - l + 1$ parts on P_i: $P_{i_1} \cdots P_{i_{r-l+1}}$ such that P_{i_j} maps onto P_{l+j-1}. It could also partition P_l and P_r in case it maps onto a sub-interval rather than covering the whole of P_l or P_r. In all, the total number of parts in the partition can increase by 0 to $n + 1$.
4. Update P so it now holds the newly induced parts as well.
5. Repeat steps $2 - 4$ until every interval P_i maps onto exactly one interval P_j already in P

If this refinement procedure terminates, we construct a graph which contains two nodes P_i^+ (direct) and P_i^- (flipped) for each interval P_i. If the post image of P_i is P_j, and the affine function on P_i preserves orientation (*i.e.*, has a positive coefficient), we connect P_i^+ to P_j^+ and P_i^- to P_j^-. If the post image of P_i is P_j, and the affine function on P_i changes orientation (*i.e.*, has a negative coefficient), we connect P_i^+ to P_j^- and P_i^- to P_j^+. We get a graph representation of the PAM. Thus, x_f is reachable from x_0, if there is a path from $P_{x_0}^+$ to $P_{x_f}^+$ in this graph (where $x_i \in P_{x_i}$), and also x_i and x_f divide their respective intervals in the same proportion $\alpha : \beta$. Another possibility is that there is a path from $P_{x_0}^+$ to $P_{x_f}^-$ in this graph, and also x_i and x_f divide their respective intervals in inverse proportions $\alpha : \beta$ and $\beta : \alpha$. $\square$

slopes $k_1 \neq k_2$) is undecidable and that the reachability problem is undecidable for rectangular automata with at least 5 clocks and one skewed clock (see LHA, section 2.2, for restrictions that have been relaxed). In [38], it was shown that the reachability question for TA with 3 stopwatches and for TA with 1 memory cell with assignments between variables is undecidable. Other undecidability results (always for the reachability problem) were given for TA with 6 memory cells without assignment between variables [39], for TA with two three-slope variables [40], for TA with two non-clock constant slope variables [10], for TA with additive clock constraints [33] and for TA with two skewed clocks [34].

Some other undecidability results were given for low (three or less) dimensional spaces, besides those mentioned results in the introduction. In [41], it was proved that Turing machines can be simulated by dynamical systems with piecewise affine functions (in 3 dimension spaces). In [20], two elementary functions are constructed: one in one dimension that simulates Turing machines with an exponential slowdown and one in two dimensions that simulate TMs in real time (see references therein for other undecidability results). Among other results, in [27], it is shown that smooth ordinary differential equations in $\mathbb{R}^2$ can simulate an arbitrary Turing machine; in [18], it is proved that TMs can be simulated by 2-dim PAMs, by 1-dim countable PAM (PAMs with an infinite number of intervals) and by a continuous piecewise-monotone functions in linear time. As a relevant result in the same work, it is also shown that there exist TMs that cannot be simulated by a 1-dim PAM. In [3], results are given concerning the frontier between decidability and undecidability for low dimensional systems, in particular for the reachability problem for 1-dim PAMs. In a more recent work, it is shown that PAMs are equivalent to Pseudo-Billiard Systems, which may be seen as 2-dim linear hybrid automata with only one state, and that more general classes of functions lead to undecidability of reachability problem for such class [42, 43].

A construction similar to the "taking-turns" idea (see section 3.2) was used by Berard and Duford to prove that the emptiness query is undecidable for timed automata with four clocks and additive clock constraints [44].

7. Conclusion

Although intense research has been pursued over the last one or two decades, there is no clear elucidation of the decidability boundary for the hybrid automaton reachability query. Fig. 1 summarizes the relationship between the main hybrid models we have considered in this work. The contribution of this paper is twofold. First, we have shown that between 2-dim PCDs (for which the reachability problem is decidable [7]) and 3-dim PCDs (for which reachability is undecidable [8]), there exists an interesting class, *2-dim HPCD*, for which the reachability question is still open. We have also shown that the same is true for several similar classes, namely 2-dim rectangular automata and 2-dim linear hybrid automata with additional restrictions, and also for PCDs on 2-dim manifolds. Moreover, we refined the decidability frontier by exploiting the expressive

redundancy of the hierarchical piecewise constant derivative system class definition. We introduced the "taking-turns" idea, that the two PCD variables could alternatively compute PAM iterations. We also showed how we could exploit the finite range of the PAM to construct non-overlapping state invariants. These ideas helped show that a 1-dim PAM can be simulated by a 2-dim PCD augmented with translational resets only, or with overlapping invariants and comparative guards, with identity resets. We also demonstrated how decidable classes, like timed and initialized rectangular automata, can be extended into classes open for the reachability problem. Second, we have proved that 2-dim HPCDs are really in the boundary between decidability and undecidability, since adding a simple counter or allowing some kind of "infinite pattern" to these systems, makes the reachability problem undecidable. A simple algorithm for over-approximating reachability was presented. It revealed that the problem is decidable, for those PAMs that converge during this iteration.

There are many related questions still unresolved: What is the least constrained 2-dim HPCD that is decidable? Can we sharpen the decidability frontier defined by *initialized rectangular HPCDs* and simple planar differential inclusion systems? Can the openness of the 1-dim PAM and the one-stopwatch automata classes be compared? A different perspective on the decidability of the PAM class may be obtained from the literature on discrete chaotic dynamical systems [45]. Another less explored research problem would be to the characterization of the undecidability problem in other models of computation, such as the Blum-Shub-Smale model [46] and models of recursive analysis.

References

[1] M. Minsky, Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines, Annals of Mathematics 74 (1961) 437–455.

[2] P. Koiran, My favourite problems, `http://www.ens-lyon.fr/~koiran/problems.html`.

[3] O. Bournez, Complexité algorithmique des systèmes dynamiques continus et hybrides, Ph.D. thesis, ENS de Lyon (1999).

[4] T. Henzinger, The theory of hybrid automata, in: LICS'96, 1996, pp. 278–292.

[5] E. Asarin, G. Schneider, Widening the boundary between decidable and undecidable hybrid systems, in: CONCUR'02, Vol. 2421 of LNCS, Springer, 2002, pp. 193–208.

[6] G. Schneider, Algorithmic analysis of polygonal hybrid systems, Ph.D. thesis, VERIMAG – UJF, Grenoble, France (July 2002).

[7] O. Maler, A. Pnueli, Reachability analysis of planar multi-linear systems, in: CAV'93, Vol. 697 of LNCS, Springer, 1993, pp. 194–209.

[8] E. Asarin, O. Maler, A. Pnueli, Reachability analysis of dynamical systems having piecewise-constant derivatives, Theoretical Computer Science 138 (1995) 35–65.

[9] V. Mysore, A. Pnueli, Refining the undecidability frontier of hybrid automata., in: FSTTCS, Vol. 3821 of LNCS, Springer, 2005, pp. 261–272.

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

[11] S. Simić, K. Johansson, S. Sastry, J. Lygeros, Towards a geometric theory of hybrid systems, in: HSCC'00, no. 1790 in LNCS, Springer, 2000, pp. 421–436.

[12] T. Henzinger, P. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, in: 27th ACM STOC, ACM Press, 1995, pp. 373–382.

[13] A. Puri, P. Varaiya, Decidability of hybrid systems with rectangular differential inclusions, in: CAV 94, no. 818 in LNCS, Springer, 1994, pp. 95–104.

[14] P. Bouyer, C. Dufourd, E. Fleury, A. Petit, Expressiveness of updatable timed automata, in: MFCS'2000, Vol. 1893 of LNCS, Springer, 2000, pp. 232–242.

[15] E. Asarin, G. Schneider, S. Yovine, On the decidability of the reachability problem for planar differential inclusions, in: HSCC'2001, Vol. 2034 of LNCS, Springer, 2001, pp. 89–104.

[16] M. Hirsch, S. Smale, Differential Equations, Dynamical Systems and Linear Algebra, Academic Press Inc., 1974.

[17] M. Henle, A Combinatorial Introduction to Topology, Dover Publications, Inc., 1979.

[18] P. Koiran, M. Cosnard, M. Garzon, Computability with low-dimensional dynamical systems, Theoretical Computer Science 132 (1) (1994) 113–128.

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

[20] P. Koiran, C. Moore, Closed-form analytic maps in one and two dimensions can simulate universal Turing machines, Theoretical Computer Science 210 (1999) 217–223.

[21] L. Bobrow, M. Arbib, Discrete Mathematics, W.B. Saunders, 1974.

[22] R. Milner, Communication and Concurrency, Prentice Hall Int., 1989.

[23] R. Devaney, An Introduction to Chaotic Dynamical Systems, 2nd Edition, Addison-Wesley, Redwood City, 1989.

[24] J. Guckenheimer, P. Holmes, Nonlinear Oscillations, Dynamical Systems and Linear Algebra, Springer, New York, 1990.

[25] K. Sibirsky, Introduction to Topological Dynamics, Noordhoff International Publishing, Leyden, 1975.

[26] E. Asarin, O. Maler, On some relations between dynamical systems and transition systems, in: ICALP'94, Vol. 820 of LNCS, Springer, 1994, pp. 59–72.

[27] M. Branicky, Universal computation and other capabilities of hybrid and continuous dynamical systems, Theoretical Computer Science 138 (1).

[28] E. Haghverdi, P. Tabuada, G. Pappas, Bisimulation relations for dynamical, control, and hybrid systems, Theoretical Computer Science 342 (2-3) (2005) 229–261.

[29] G. Teschl, Ordinary differential equations and dynamical systems, Lecture Notes from `http://www.mat.univie.ac.at/~gerald/ftp/book-ode/index.html` (2004).

[30] P. Bouyer, C. Dufourd, E. Fleury, A. Petit, Are timed automata updatable?, in: CAV'2000, Vol. 1855 of LNCS, Springer, 2000, pp. 464–479.

[31] R. Alur, S. Kannan, S. L. Torre, Polyhedral flows in hybrid automata., in: HSCC'99, no. 1569 in LNCS, Springer, 1999, pp. 5–18.

[32] V. Mysore, B. Mishra, Algorithmic algebraic model checking III: Approximate methods, in: Infinity'05, Vol. 149 of ENTCS, 2006, pp. 61–77.

[33] R. Alur, D. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.

[34] R. Alur, C. Courcoubetis, T. Henzinger, P.-H. Ho, Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems, in: Hybrid Systems, Vol. 736 of LNCS, Springer, 1993, pp. 209–229.

[35] X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, An approach to the description and analysis of hybrid systems, in: Hybrid Systems, Vol. 736 of LNCS, Springer, 1993, pp. 149–178.

[36] A. Bouajjani, R. Robbana, Verifying omega-regular properties for a subclass of linear hybrid systems, in: CAV'95, no. 939 in LNCS, 1995, pp. 437–450.

[37] A. Bouajjani, R. Echahed, R. Robbana, Verifying invariance properties of timed systems with duration variables, in: FTRTFT'94, no. 863 in LNCS, 1994, pp. 193–210.

[38] K. Čerāns, Algorithmic problems in analysis of real-time systems specifications, Ph.D. thesis, Univ. of Latvia (1992).

[39] R. Alur, T. Henzinger, M. Vardi, Parametric real-time reasoning, in: 25th ACM STOC, 1993, pp. 592–601.

[40] Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine, Integration graphs: a class of decidable hybrid systems, in: Hybrid Systems, Vol. 736 of LNCS, Springer, 1993, pp. 179–208.

[41] C. Moore, Unpredictability and undecidability in dynamical systems, Physical Review Letters 64 (20).

[42] O. Kurganskyy, I. Potapov, F. Sancho-Caparrini, Computation in one-dimensional piecewise maps, in: HSCC'07, Vol. 4416 of LNCS, Springer, 2007, pp. 706–709.

[43] O. Kurganskyy, I. Potapov, F. Sancho-Caparrini, Reachability problems in low-dimensional iterative maps, Int. J. Found. Comput. Sci. 19 (4) (2008) 935–951.

[44] B. Berard, C. Dufourd, Timed automata and additive clock constraints, Information Processing Letters 75 (1-2) (2000) 1–7.

[45] E. Asarin, Chaos and undecidability, Tech. rep., Verimag (1995).

[46] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer, 1997.