Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives*

Eugene Asarin † Oded Maler ‡ Amir Pnueli§

November 10, 1994

*This research was supported in part by the France-Israel project for cooperation in Computer Science, by the European Community ESPRIT Basic Research Action Projects REACT (6021) and by Research Grant #01-00884 of the Russian Foundation of Fundamental Research

†Institute for Information Transmission Problems, 19 Ermolovoy st., Moscow, Russia, E-mail: asarin@ippi.ac.msk.su . During the preparation of the paper the author was visiting the University of Paris 12, Val de Marne.

‡SPECTRE-VERIMAG, Miniparc-ZIRST, 38330 Montbonnot, France, Oded.Maler@imag.fr . (VERIMAG is a joint laboratory of CNRS, INPG, UJF and VERILOG SA. SPECTRE is a project of INRIA).

§Department of Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: amir@wisdom.weizmann.ac.il .

Abstract

In this paper we consider a class of hybrid systems, namely dynamical systems with piecewise-constant derivatives (PCD systems). Such systems consist of a partition of the Euclidean space into a finite set of polyhedral sets (*regions*). Within each region the dynamics is defined by a constant vector field, hence discrete transitions occur only on the boundaries between regions where the trajectories change their direction.

With respect to such systems we investigate the reachability question: *Given an effective description of the systems and of two polyhedral subsets P and Q of the state-space, is there a trajectory starting at some $\mathbf{x} \in P$ and reaching some point in Q?* Our main results are a decision procedure for two-dimensional systems, and an undecidability result for three or more dimensions.

1 Introduction

1.1 Motivation

Hybrid systems (HS) are systems that combine intercommunicating discrete and continuous components. Most embedded systems belong to this class since they operate and interact with a continuous environment, and are expected to provide real-time responses to continuously varying situations.

The introduction of HS models is motivated by a real practical concern: with the decrease in the size and price of computing elements, more and more computers (discrete state-transition systems) are embedded within real-world control loops such as in avionics, process control, robotics and consumer products – to mention a few application areas. The analysis and prediction of the combined behavior of these embedded systems require formal tools that cut across existing disciplinary boundaries: the real-world is usually modeled by control engineers as a continuous dynamical system while computer scientists investigate the dynamics of discrete systems.

The ultimate goal of the theory of HS is to build models of such embedded systems, models which include the dynamics of an external environment and the interface between the controller and the environment. Within these models, based upon a description of a discrete controller (such as a program or a digital circuit) and upon its timing characteristics, it will be possible to prove that the behavior of the controlled environment satisfies certain properties. Even if we cannot realistically hope for fully algorithmic analysis techniques, any progress along this line of research will enhance the quality of current design methodologies, and will provide system developers with models and with software tools that will result in a more efficient, systematic and reliable development process.

1.2 Models for Hybrid Systems

Hybrid systems generalize both discrete state-transition systems and continuous dynamical systems. A HS consists of two types of state-variables: discrete variables whose values change via discrete state-transitions, and continuous variables which change continuously according to some dynamical law during the interval between two consecutive state-transitions. These two types of variables interact with each other in the following ways:

1. Some property satisfied by continuous variables (e.g., a variable crosses a threshold) enables or disables a discrete state-transition.

2. A change in a discrete variable may change the dynamical law to which some continuous variables are subject.

The first formal model in the verification literature linking continuous and discrete dynamics was the *phase transition systems* introduced in [13]. Several, more or less similar, models for hybrid systems have been proposed and investigated recently (see, for example, [16], [4]). Various negative and positive results concerning the decidability of verification problems in these models have been established. The positive results usually involved the special case of timed automata [1], [3], [10], whose introduction was motivated by real-time systems. Timed automata can be viewed as hybrid systems where the only continuous variables are timers, all varying at the same rate, and possibly being reset by discrete transitions. Tests on values of those timers serve as "guards" for performing the discrete transitions – usually those tests are conjunctions of simple linear inequalities in one variable, or linear inequalities on the difference between two clocks.

Similar to [12] (Integration Graphs), we are looking for decidable subclasses of Hybrid Systems with piecewise-constant derivatives (PCD systems). The strategy taken in [12] is based on placing restrictions on the guards of transitions that may occur within loops. Here we take an alternative approach and allow arbitrary boolean combinations of linear inequalities as transition guards, but place the following restrictions:

- The system is deterministic (this restriction is somewhat relaxed toward the end of the paper).
- Transitions do not modify any of the continuous variables. Thus, there are no resets or other assignment statements.
- The control component of a state is determined by the values of the continuous variables. Thus, if the corresponding PCD system is represented as a state-graph, it is not possible to reach different nodes (possibly in different computations) with the same set of values for the continuous variables.

Using the terminology of [16], the guards for all the transitions outgoing from a given location are mutually exclusive and their union is the complement of the invariant condition for that location.

Due to the fact that the discrete (control) component of the state is fully determined by the values of the continuous variables and that all changes are continuous, we prefer to present the system without explicit reference to discrete states. Instead, the system is viewed as a set of *regions* (corresponding to discrete states in other presentations) with boundaries separating them. Each region is associated with a constant *vector field* which identifies the rates at which the various variables change. Reaching a boundary and crossing into another region is equivalent to taking a transition to another discrete state in which the continuous evolution rule is different.

Compared to classes of hybrid and real-time systems considered so far in the verification literature, the model investigated in this paper is in some aspects, more general and in some aspects more restrictive. On one hand, we drop the restriction of a uniform slope for all continuous variables, assumed in timed automata, and allow each variable to have its own slope within a discrete state. We also allow the continuous variables to appear in more general guards for discrete transitions (combinations of arbitrary linear equalities). On the other hand, our class of systems is more restrictive, mainly because we do not allow discrete "jumps" in the values of the variables (such as those caused by assignment statements). Thus the trajectories of the system are continuous, but not smooth. Systems obeying such requirement are closer to continuous dynamical systems and are more amenable to topological and geometrical analysis. Some of the recent research in hybrid systems, may appear to be too dominated either by continuous or by computer science techniques, depending on the authors' origin. The model underlying this paper can be seen as an attempt to balance the situation by generalizing the continuous dynamics and simplifying the discrete component.

1.3 An Example: Hunter and the Hunted

In order to motivate the reader we will present a toy problem which is analyzable using the techniques developed in this paper.

The problem involves two players which move in an one-dimensional space. Their respective positions are denoted by the state-variables x and y and we assume that initially $x < y$. When x and y occupy certain positions, party y is the pursuer while party x flees away from y. However, when x and y occupy different positions, the roles may be reversed and x may assume the role of a pursuer while y flees away. Such a situation arises, for example, in the video game *Pacman* in which, under normal circumstances, the Pacman is pursued by one or more ghosts. However, when the Pacman eats a certain fruit, the roles are reversed and the Pacman starts chasing the ghost. We therefore may consider the position of the Pacman and its pursuing ghost to be given by x and y, respectively.

The behavior of each of the players consists of running either left or right at certain velocities depending on the relative location of their opponents. The ghost runs to the left at velocity b_1 when $x < 0$ and runs to the right at velocity b_2 when $x > 0$. The Pacman runs to the left at velocity a_1 when $y > 2$ and runs to the right at velocity a_2 when $y < 2$.

The configuration of the two players is displayed in figure 1. The precise behavioral rules of Pacman and the ghost are depicted in table 1. All the parameters are positive and the entries in the table denote velocities.

Knowing all the parameters of the system we would like to answer questions such as: Given that Pacman starts at some position in the interval $[x_0, x_1]$, and that the ghost starts at some position in $[y_0, y_1]$, will they ever meet ($x = y$)? Is it possible that the distance between them will become larger than some d^*? Will Pacman ever reach some point x^*?

If we look at the positions of Pacman and the ghost as the coordinates

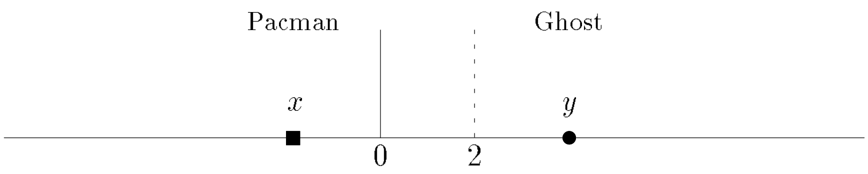


Figure 1: Pacman x and the ghost y.

	$x > 0,\ y > 2$	$x < 0,\ y > 2$	$x < 0,\ y < 2$	$x > 0,\ y < 2$
V_{Pacman}	$-a_1$	$-a_1$	a_2	a_2
V_{Ghost}	b_2	$-b_1$	$-b_1$	b_2

Table 1: The behavioral rules of Pacman and the ghost.

of our system, we obtain a planar (2-dimensional) PCD system (see exact definitions below). Each point in the x, y plane represents the joint positions of Pacman and the ghost. Their corresponding rules of behavior induce a partition of the plane into regions such that within every region the system evolves with a constant slope of the form $\mathbf{c} = (V_{Pacman}, V_{Ghost})$ as depicted in figure 2. In this paper we show how for every system of this type, reachability questions between polyhedral subsets of the state-space can be effectively answered. On the other hand we show that for 3-dimensional systems (e.g., by adding a third player to the game) there is no general reachability algorithm.

The rest of the paper is organized as follows: In section 2 we introduce the necessary geometrical and topological prerequisites and define the class of systems we are dealing with. In section 3 we present planar systems and prove some of their important properties which are crucial for the termination of our decision procedure. The computational framework for forward simulation is developed in section 4, culminating in the decision procedure for reachability between points, which is extended in section 5 to reachability between regions. In section 6 we demonstrate the computational power of PCD systems and show that three dimensions are sufficient for simulating two-stack machines, and hence the reachability problem for 3-dimensional systems is undecidable.

The paper is a combination and elaboration of results presented in [14] and [2].

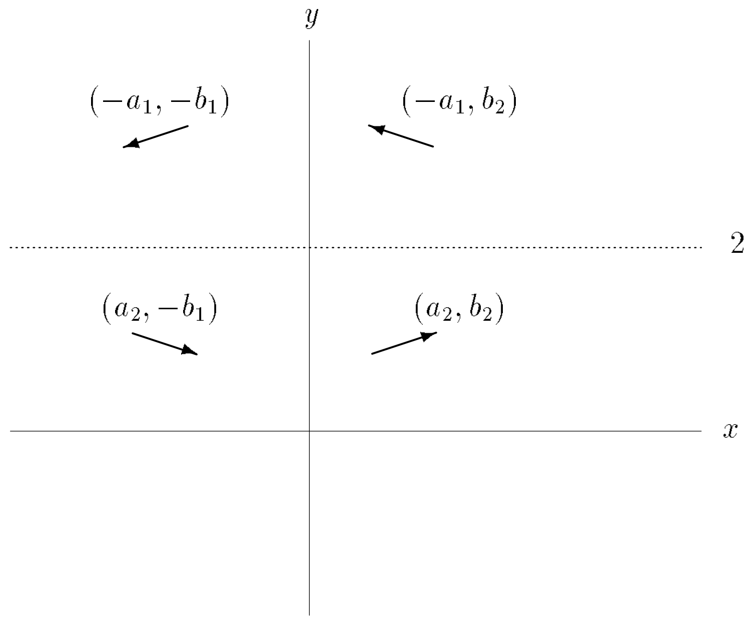


Figure 2: Pacman x and the ghost y viewed as a planar PCD system.

2 Preliminaries

Throughout this paper, we deal with the d-dimensional Euclidean space[1] $X = \mathbb{R}^d$. Points (vectors) in X are denoted by boldface letters such as $\mathbf{x}$ or $\mathbf{a}$. The expression $\mathbf{a} \cdot \mathbf{x}$ denotes the standard inner product. For $\mathbf{x}_1, \ldots \mathbf{x}_n \in X$ an affine combination is $\mathbf{x} = \sum \lambda_i \mathbf{x}_i$ such that $\sum \lambda_i = 1$. A convex combination is the same with $\lambda_i > 0$ for every i. The affine (resp. convex) hull of a set $A \subseteq X$ is the set of all affine (resp. convex) combinations of points in A. They are denoted by $\mathit{aff}(A)$ and $\mathit{conv}(A)$. A subset P of X is convex if $P = \mathit{conv}(P)$. The dimension of P, $\mathit{dim}(P)$ is the dimension of $\mathit{aff}(P)$. Intuitively, the affine hull is the generalization of concepts such as "the line connecting two points" or "the plane induced by 3 non-co-linear points", etc.

The interior $\mathit{int}(P)$ of $P \subseteq X$ is the set of points $\mathbf{x} \in P$ such that there exist a neighborhood $N_\epsilon(\mathbf{x}) \subseteq X$ such that $N_\epsilon(x) \subseteq P$. The boundary of P is $\mathit{bd}(P) = \mathit{cl}(P) - \mathit{int}(P)$ where $\mathit{cl}(P)$ denotes closure. If P is e-

[1]A readable introduction to convex and polyhedral sets can be found in [7].

dimensional for some $e < d$ then it has no interior points because every neighborhood in X^d is d-dimensional and thus contains parts outside P. On the other hand we can define the notion of relative-interior by relaxing the above condition into $(N_\epsilon(x) \cap \mathit{aff}(P)) \subseteq P$. The set of all relative interior points of P is denoted by $ri(P)$. Similarly the relative boundary of P is defined as $rb(P) = cl(P) - ri(P)$. For example a closed segment in a 2-dimensional space has no interior points, but its "open" sub-segment is its relative interior and its endpoints constitute its relative boundary.

An open (closed) *half-space* in X is the set of all points $\mathbf{x} \in X$ satisfying $\mathbf{a}\cdot\mathbf{x}+b < 0$ ($\mathbf{a}\cdot\mathbf{x}+b \leq 0$). A *convex polyhedral set* is an intersection of finitely many half-spaces. A *time segment* is any interval $[0, r] \subseteq \mathbb{R}^+$ including $\mathbb{R}^+$ itself. A *trajectory* in X is a continuous function $\xi : T \to X$ where T is a time segment.

Definition 1 (Dynamical System) *An (autonomous) dynamical system is $\mathcal{H} = (X, f)$ where X is the state-space and f is a partial function from X to X such that $\frac{d^+\mathbf{x}}{dt} = f(\mathbf{x})$ is the differential equation governing the evolution of $\mathbf{x}$. A trajectory of $\mathcal{H}$ starting at some $\mathbf{x}_0 \in X$ is $\xi : T \to X$ such that $\xi()$ is a solution of the equation with initial condition $\mathbf{x} = \mathbf{x}_0$, i.e., $\xi(0) = \mathbf{x}_0$ and for every t, $f(\xi(t))$ is defined and is equal to the right derivative of $\xi(t)$.*

A differential equation has a uniqueness property if for every $\mathbf{x}_0$ there is at most one solution. In this case the system is said to be deterministic. This paper treats a sub-class of dynamical systems, namely those having piecewise-constant (possibly discontinuous) derivatives:

Definition 2 (PCD System) *A piecewise-constant derivative (PCD) system is a dynamical system $\mathcal{H} = (X, f)$ where f is a (possibly partial) function from X to X such that the range of f is a finite set of vectors $C \subset X$, and for every $\mathbf{c} \in C$, $f^{-1}(\mathbf{c})$ is a finite union of convex polyhedral sets.*

In other words, a PCD system consists of partitioning the space into convex polyhedral sets ("regions"), and assigning a constant derivative $\mathbf{c}$ ("slope") to all the points sharing the same region. The trajectories of such systems are broken lines, with the breakpoints occurring on the boundaries of the regions. The example in the introduction (figure 2) is a PCD system.

A description of a PCD system is simply a list of the regions (expressed as intersections of linear inequalities) and their corresponding slope vectors.

From now on we assume that all the constants in the system's definition are rational. Note that, unlike more general dynamical systems, PCD systems are effective in the following sense: Given a description of the system, and a rational initial point $\mathbf{x}$, there exists some positive $\epsilon > 0$ such that, for every Δt, $0 < \Delta t < \epsilon$, one can calculate *precisely* the point $\mathbf{x}'$ which a trajectory starting at $\mathbf{x}$ will reach after time Δt.

Given a description of a PCD system $\mathcal{H}$, the reachability problem for $\mathcal{H}$, denoted by $\mathbf{Reach}(\mathcal{H}, \mathbf{x}, \mathbf{x}')$ is the following: *Given* $\mathbf{x}, \mathbf{x}' \in X$*, are there a trajectory* ξ *and* $t \geq 0$ *such that* $\xi(0) = \mathbf{x}$ *and* $\xi(t) = \mathbf{x}'$*?* The region-to-region reachability problem $\mathbf{R\text{-}Reach}(\mathcal{H}, P, P')$ is: *Given two polyhedral sets* $P, P' \subseteq X$*, are there two points* $\mathbf{x} \in P$ *and* $\mathbf{x}' \in P'$ *such that the answer to* $\mathbf{Reach}(\mathcal{H}, \mathbf{x}, \mathbf{x}')$ *is positive?*

3 Planar Systems and their Properties

In the following sections we will concentrate on planar PCDs, i.e., $X = I\!R^2$, with the additional restriction that all the regions are 2-dimensional. For every vector $\mathbf{x} = (x_1, x_2) \in I\!R^2$ we define its *right rotation* as the vector $\hat{\mathbf{x}} = (x_2, -x_1)$. Clearly $\mathbf{x} \cdot \hat{\mathbf{x}} = 0$ and $\mathbf{x} \cdot \mathbf{y} = \hat{\mathbf{x}} \cdot \hat{\mathbf{y}}$.

Definition 3 (Polyhedral partition) *A finite polyhedral partition of* X *is a family* $\mathcal{P} = \{P_1, \ldots, P_k\}$ *of open full-dimensional polyhedral sets such that* $\bigcup_{i=1}^{k} cl(P_i) = X$ *and for every* $P_i, P_j \in \mathcal{P}$*,* $P_i \cap P_j = \emptyset$*.*

We will denote by $bd(\mathcal{P})$ the set of all points in X which are in $bd(P)$ for some $P \in \mathcal{P}$ and by $E(\mathcal{P})$ the set of *edges* of $\mathcal{P}$, namely non-empty subsets of X of the form $e = ri(cl(P_i) \cap cl(P_j))$ for some $P_i, P_j \in \mathcal{P}$. Similarly the set of *vertices* of $\mathcal{P}$, $V(\mathcal{P})$ consists of points $\mathbf{x} \in X$ such that $\{\mathbf{x}\} = cl(e_i) \cap cl(e_j)$ for some $e_i, e_j \in E(\mathcal{P})$. We call the elements of $B(\mathcal{P}) = E(\mathcal{P}) \cup V(\mathcal{P})$ *boundary elements*. One can easily see that X is decomposed into a disjoint union $\mathcal{P} \cup E(\mathcal{P}) \cup V(\mathcal{P})$. For example, the polyhedral partition in figure 3 has 5 regions, 3 vertices and 7 edges.

Suppose e is an edge such that $e \subseteq bd(P) \cap bd(P')$. Let $P = \{\mathbf{x} : \bigwedge_{i \in I} \mathbf{a}_i \cdot \mathbf{x} + b_i < 0\}$ and $P' = \{\mathbf{x} : \bigwedge_{i' \in I'} \mathbf{a}_{i'} \cdot \mathbf{x} + b_{i'} < 0\}$. Then for e to be non-empty there must be some $j \in I$, $k \in I'$, such that $\mathbf{a}_j = -\mathbf{a}_k$ and $b_j = -b_k$ and every $\mathbf{x} \in e$ satisfies $\mathbf{a}_j \cdot \mathbf{x} + b_j = 0$. We call $\mathbf{a}_j$ and $\mathbf{a}_k$ the

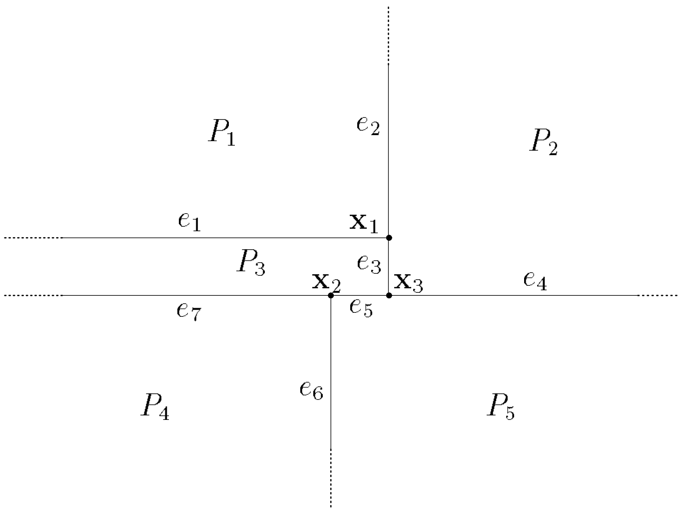


Figure 3: *A polyhedral partition of the plane.*

characteristic vectors of e relative to P and P'. One can see that they are two opposite normals to e.

Definition 4 (Planar PCD systems) *A planar PCD system is given by* $\mathcal{H} = (\mathcal{P}, \varphi, \psi)$ *where* $\mathcal{P}$ *is a polyhedral partition of* $X = \mathbb{R}^2$, $\varphi : \mathcal{P} \to X$ *is a function which assigns to each region a slope vector in* X, *and a function* $\psi : B(\mathcal{P}) \to \mathcal{P}$ *satisfying* $b \subseteq cl(\psi(b))$ *for every boundary element (edge or vertex)* $b \in B(\mathcal{P})$.

The function ψ simply associates every boundary element with one of its neighboring regions. One can easily see that by letting $f(\mathbf{x}) = \varphi(P)$ when $\mathbf{x} \in P \in \mathcal{P}$ and $f(\mathbf{x}) = \varphi(\psi(b))$ when $\mathbf{x} \in b \in B(\mathcal{P})$ we obtain the more general definition 2. We denote $\varphi(P_i)$ by $\mathbf{c}_i$.

Orientation and Ordering of Boundaries

For $b \in B(\mathcal{P})$ such that $\varphi(P) = \mathbf{c}$, we say that b is an *entry boundary element* of P if for every $\mathbf{x} \in b$, we have $\{\mathbf{x} + \mathbf{c}t : 0 < t < \epsilon\} \subseteq int(P)$ for some $\epsilon > 0$. This implies that the vector $\mathbf{c}$ taken at any entry boundary point of P points into the interior of P. Similarly b is an exit element if the same condition holds for some $\epsilon < 0$ and for all t, $\epsilon < t < 0$.

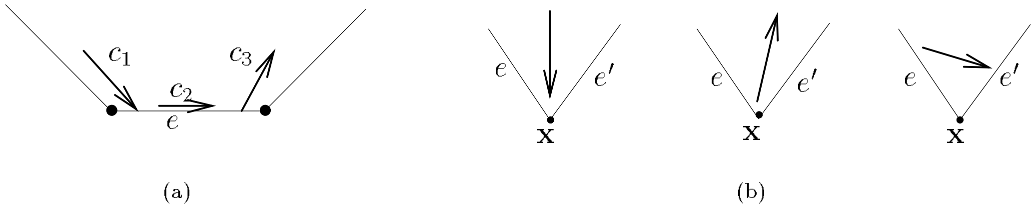


Figure 4: *The possible orientations between a region and its borders: (a) e is an exit, a tangent or an entry to P according to whether $\varphi(P)$ is $\mathbf{c}_1$, $\mathbf{c}_2$ or $\mathbf{c}_3$ respectively. (b) $\mathbf{x}$ is an exit, an entry, or neutral with respect to P.*

Note that we may have an edge e and a vertex $\mathbf{x} \in cl(e)$ such that $\psi(e) \neq \psi(\mathbf{x})$. For example, in the partition of figure 3, we may have $\psi(e_1) = P_3$, while $\psi(\mathbf{x}_1) = P_2$.

Consider a region P and one of its edges e with a characteristic vector $\mathbf{a}$ relative to P (i.e., $\mathbf{a}$ points from e into P), and let $\mathbf{c} = \varphi(P)$. Then, by simple calculation one can see that e is an *entry* to P iff $\mathbf{a} \cdot \mathbf{c} > 0$, e is an *exit* from P iff $\mathbf{a} \cdot \mathbf{c} < 0$, and e is neutral (neitehr entry nor exit) if $\mathbf{a} \cdot \mathbf{c} = 0$. For simplicity of presentation we assume that the system is not degenerate, i.e., no edge is neutral with respect to a neighboring region. If e is on the boundaries of P_i and P_j, it is required that e be an entry to one of them (say P_i) and an exit to the other, and be affiliated with the region to which it is an entry, i.e., $\psi(e) = P_i$. From now on, we adopt the convention that the characteristic vector of an edge e is the one pointing into the region $P = \psi(e)$, i.e., the region to which e is an entry edge.

For every vertex $\mathbf{x} \in bd(P)$ there are exactly two edges $e, e' \subseteq bd(P)$ such that $\mathbf{x} = cl(e) \cap cl(e')$. Then one can see that $\mathbf{x}$ is an entry point to P if both e and e' are entry edges. Symmetrically it is an exit point if both e and e' are exit edges. Otherwise, when one of $\{e, e'\}$ is an entry edge and the other is an exit edge we say that $\mathbf{x}$ is neutral w.r.t. P. We require that $\psi(\mathbf{x}) = P$ iff $\mathbf{x}$ is an entry point of P. Thus we have completed the classification of all boundary points according to their orientation relative to the region (see figure 4). We will denote the set of all entry points of P by $In(P)$ and set of all exit points by $Out(P)$.

Next we prove some fundamental properties of planar systems which apply to an even more general class than the PCD systems considered in this paper. A sufficient condition for these properties to hold is that any straight

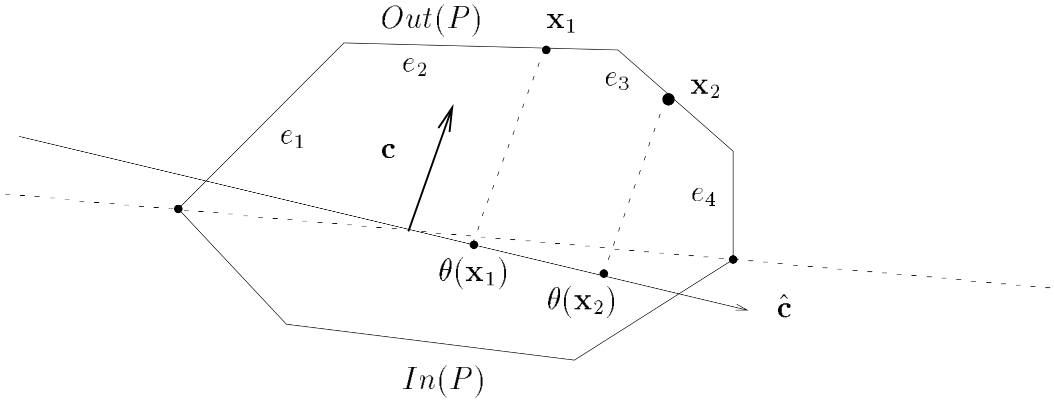


Figure 5: *Ordering on the boundary:* $\mathbf{x}_1 \preceq \mathbf{x}_2$.

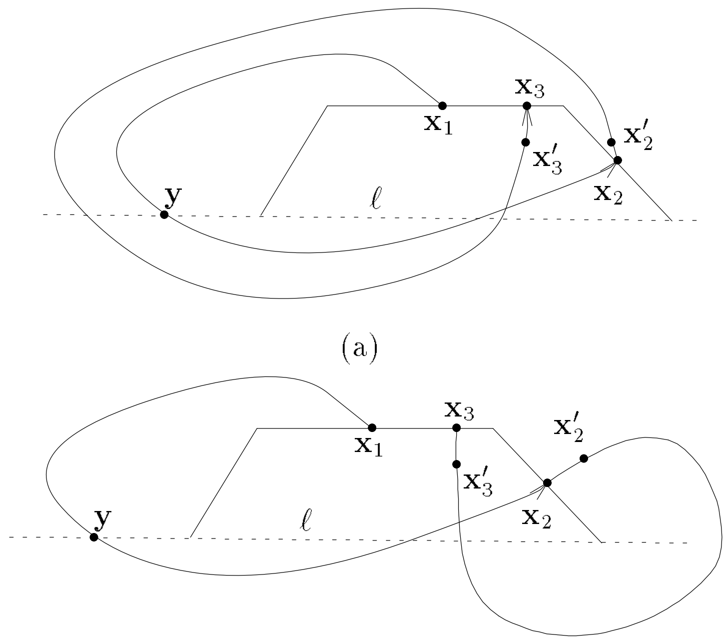
line can be divided into finitely many segments, each of which can be traversed by any trajectory in at most one direction.

Consider a region P with $\varphi(P) = \mathbf{c}$ whose boundary is partitioned into $In(P)$ and $Out(P)$. The mapping $\theta : X \to I\!R$, defined as $\theta(\mathbf{x}) = \mathbf{x} \cdot \hat{\mathbf{c}}$, assigns to every $\mathbf{x} \in X$ a value proportional to the length of the projection of the vector $\mathbf{x}$ on the right rotation of $\mathbf{c}$. One can easily see that the relation $\preceq$, defined as $\mathbf{x}_1 \preceq \mathbf{x}_2$ if $\theta(\mathbf{x}_1) \leq \theta(\mathbf{x}_2)$, is a dense linear order on $In(P)$ and $Out(P)$ (see figure 5).

The fact that $In(P)$ and $Out(P)$ are ordered allows us to speak of *boundary intervals* of the form $[\mathbf{x}_1, \mathbf{x}_2]$ denoting all the points $\mathbf{x} \in In(P)$ (or $Out(P)$) satisfying $\mathbf{x}_1 \preceq \mathbf{x} \preceq \mathbf{x}_2$. We use $\prec$ to denote the strict variant of $\preceq$ and say that $e_1 \prec e_2$ if $\mathbf{x}_1 \prec \mathbf{x}_2$ for every $\mathbf{x}_1 \in e_1, \mathbf{x}_2 \in e_2$. For example, in figure 5 we have $e_1 \prec e_2 \prec e_3 \prec e_4$.

Claim 1 (Fundamental property of planar systems) *Let ξ be any trajectory that intersects $In(P)$ (or $Out(P)$) in three consecutive points,* $\mathbf{x}_1$*,* $\mathbf{x}_2$ *and* $\mathbf{x}_3$*. Then,* $\mathbf{x}_1 \preceq \mathbf{x}_2$ *implies* $\mathbf{x}_2 \preceq \mathbf{x}_3$ *and* $\mathbf{x}_1 \succeq \mathbf{x}_2$ *implies* $\mathbf{x}_2 \succeq \mathbf{x}_3$*.*

Proof: If $\mathbf{x}_1 = \mathbf{x}_2$ then due to determinism $\mathbf{x}_2 = \mathbf{x}_3$. So we assume $\mathbf{x}_1 \prec \mathbf{x}_2$ and show that $\mathbf{x}_3 \prec \mathbf{x}_2$ implies an intersection of ξ with itself in the segment between $\mathbf{x}_2$ and $\mathbf{x}_3$, which contraditcs $\mathbf{x}_3 \prec \mathbf{x}_2$. Let ℓ be some line separating $In(P)$ and $Out(P)$. Suppose, without loss of generality, that the trajectory from $\mathbf{x}_1$ to $\mathbf{x}_2$ circumvents P from the left (i.e., its last intersection with line ℓ before re-entering P is some $\mathbf{y} \prec \mathbf{x}_1$) (see figure 6). Consider now



(b)

Figure 6: *Illustration of the fundamental property: (a) Circumvention from the left. (b) Circumvention from the right.*

the closed set S bounded by the closed curve consisting of the trajectory from $\mathbf{x}_1$ to $\mathbf{x}_2$ and the boundary interval $[\mathbf{x}_1, \mathbf{x}_2]$. In order that there will be a trajectory from $\mathbf{x}_2$ to $\mathbf{x}_3$ there must be two points, $\mathbf{x}_2'$ "above" $\mathbf{x}_2$ and $\mathbf{x}_3'$ "below" $\mathbf{x}_3$ such that there is a trajectory $\mathbf{x}_2 \to \mathbf{x}_2' \to \mathbf{x}_3' \to \mathbf{x}_3$ and in particular we can choose $\mathbf{x}_2'$ outside S and $\mathbf{x}_3'$ inside S (if we cannot, then $\mathbf{x}_2$ and $\mathbf{x}_3$ coincide). Consequently, according to Jordan's theorem, the trajectory $\mathbf{x}_2' \to \mathbf{x}_3'$ must intersect the boundary of S. Since it cannot do it on $[\mathbf{x}_1, \mathbf{x}_2]$, ξ must intersect itself and this contradicts determinism unless $\mathbf{x}_2 = \mathbf{x}_3$. This is true independent of whether the trajectory from $\mathbf{x}_2$ to $\mathbf{x}_3$ circumvents P from the left (figure 6-a), or from the right (figure 6-b). ∎

This topological property has many consequences concerning the set of possible trajectories. It implies that the sequence of consecutive intersection points of a trajectory with $In(P)$ or $Out(P)$ is monotone with respect to $\preceq$. In fact, the relation between $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{y}$ determines two classes of possible

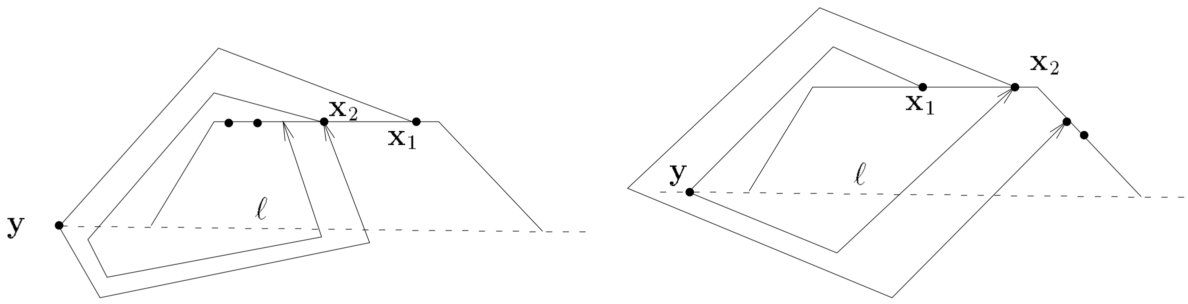


Figure 7: *(a) A contracting spiral. (b) An expanding spiral.*

"quasi-cycles": if $\mathbf{x}_2$ is between $\mathbf{x}_1$ and $\mathbf{y}$ we have a "contracting spiral" (fig 7-a) while if $\mathbf{x}_1$ is between $\mathbf{x}_2$ and $\mathbf{y}$ we have an "expanding spiral" (fig 7-b). Once a trajectory has performed a contracting spiral, it will never reach any point "outside" the spiral (i.e., outside the closed curve formed by the trajectory $\mathbf{x}_1 \to \mathbf{y} \to \mathbf{x}_2$ and the boundary interval $[\mathbf{x}_1, \mathbf{x}_2]$) and, symmetrically, after performing an expanding spiral, a trajectory will never reach a point "inside" the spiral.

Traces and Signatures

Let $\xi : T \to X$ be trajectory defined on a polyhedral partition $\mathcal{P}$. The *trace* of ξ is the sequence $\bar{\xi} = \mathbf{x}_1, \mathbf{x}_2, \ldots$ of the intersection points of ξ with $B(\mathcal{P})$. We use the notation $\bar{\xi}_{[i..j]}$ to denote the subsequence $\mathbf{x}_i, \ldots, \mathbf{x}_j$ of $\bar{\xi}$. We say that $\mathbf{x}_{i+1}$ is the *immediate successor* of $\mathbf{x}_i$.

Note that the finiteness of $\bar{\xi}$ is not necessarily determined by the boundedness of the time interval T: a trajectory ξ defined over the whole $\mathbb{R}^+$ may reach some "terminal" region and hence will stop crossing boundaries and $\bar{\xi}$ will be a finite sequence. On the other hand, a contracting spiral can cross boundaries infinitely many times during a finite real-time interval.

The qualitative behavior of a trajectory ξ such that $\bar{\xi} = \mathbf{x}_1, \mathbf{x}_2, \ldots$, is captured by its *signature*, which is the sequence of boundary elements (edges and vertices), $\tau(\xi) = b_1, b_2, \ldots$ such that for every i, $\mathbf{x}_i \in b_i$. The sequence $\rho(\xi) = P_1, P_2, \ldots$ is the corresponding *region signature* of ξ satisfying the obvious relation $b_i \subseteq In(P_i)$. Since vertices can be seen as a degenerate case of edges, we will henceforth refer to signatures as to sequences of *edges*.

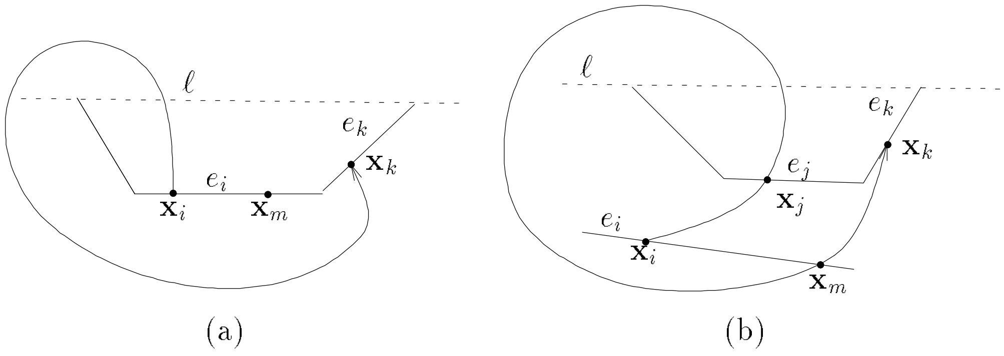


Figure 8: *Irreversibility of abandonment.*

A sequence $w = w_1, \ldots, w_n$ over some finite alphabet is called a (primitive) *cycle* if $w_1 = w_n$ and $\{w_i, \ldots, w_{n-1}\}$ are pairwise different. A trace $\bar{\xi}_{[i..j]} = \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j$ forms a *region cycle* if its corresponding region signature, $P_i, \ldots, P_j$ is a cycle. It forms an *edge cycle* if its signature $e_i, \ldots, e_j$ is a cycle. Note that if $\bar{\xi}_{[i..j]}$ is an edge cycle it is also a region cycle but not vice versa.

An edge e is said to be *abandoned* by a trajectory after position i, if $e_i = e$ and for some j, k, $i \leq j < k$, $\bar{\xi}_{[j..k]}$ forms a region cycle and $e \notin \{e_{i+1}, \ldots, e_k\}$.

Claim 2 (Abandonment is Irreversible) *An edge e abandoned after position i will not appear in the signature at any position $m > i$.*

Proof: Let $\bar{\xi}_{[j..k]}$ be the trace that forms the relevant region cycle. We assume without loss of generality that $\mathbf{x}_j \prec \mathbf{x}_k$ and that the cycle circumvents P_j from the left (expanding spiral). Consider the case of $i = j$. Then, another intersection of ξ with e_i must happen at some $\mathbf{x}_m \prec \mathbf{x}_k$ and this will violate the monotonicity property (see figure 8-a). For the case $i < j$, e_i is an edge not belonging to $In(P_j)$ and $\mathbf{x}_i$ must be inside the set enclosed by the trajectory from $\mathbf{x}_j$ to $\mathbf{x}_k$. If e_i is completely contained within this set, it cannot be reached from $\mathbf{x}_k$ without self-intersection of ξ. If e_i is not fully contained, then it must intersect the trajectory from $\mathbf{x}_j$ to $\mathbf{x}_k$ and will appear in the signature at some position m, $j < m < k$ and contradict the assumption of abandonment (see figure 8-b). A similar argument holds for the case of a contracting spiral. ∎

Corollary 3 *Every trajectory has an ultimately-periodic signature, i.e., a signature of the form* $e_1, \ldots, e_i, (e_{i+1}, \ldots, e_{i+j})^\omega$ *for some finite* i, j *where* j *is at most the number of regions.*

Proof: Because of the finite number of edges every infinite signature contains a subset E of edges appearing infinitely often. Due to claim 1, this subset may contain *at most* one entry edge per region. Let i be a position in the signature after all elements of E have already occurred and all the elements of $E(\mathcal{P}) - E$ have already disappeared. Let $e = e_i$, then the remaining signature can be factorized into $e, \sigma_1, e, \sigma_2, e \ldots$ where $\sigma_j \in (E - \{e\})^*$ for every $j > 0$. All the elements of $E - \{e\}$ must appear in every σ_j, otherwise they are abandoned. In addition every σ_j is cycle-free, otherwise e must be abandoned. Finally it is impossible for an edge e' to appear before e'' in σ_j and after e'' in σ_{j+1} as in the sequence

$$e \ldots e' \ldots, e'' \ldots, e \ldots, e'', \ldots, e', \ldots, e$$

because otherwise e' will be abandoned due to the cycle $e'' \ldots, e \ldots, e''$. Consequently, all the σ_j must be identical and the sequence is ultimately-periodic. ∎

Equipped with this nice qualitative property (which does not hold in higher dimensions), we turn to the actual calculation of trajectories of a given PCD system.

4 Point-to-point Reachability

In this section we devise a framework for representing edges and vertices that will provide for the exact characterization and calculation of the set of points reachable by a trajectory starting at a given point.

Definition 5 (Representations) *Let* $e \in In(P)$ *be an edge with characteristic vector* $\mathbf{a}$ *(pointing into* P*). A representation for* e *consists of two vectors* $\mathbf{v} \in cl(e)$*,* $\mathbf{u} = \hat{\mathbf{a}}$ *(the right rotation of* $\mathbf{a}$*), and two numbers* l, h*, such that* l *is a rational or the special value* $-\infty$*,* $h > l$ *is a rational or* ∞*, and*

$$e = \{\mathbf{v} + \lambda \mathbf{u} : l < \lambda < h\}$$

The choice of $\mathbf{v}$ implies, of course, the values of l and h. Having fixed $\mathbf{v}$ and $\mathbf{u}$ for every edge we can uniquely represent every non-vertex boundary point $\mathbf{x}$ by a pair (e, λ) identifying the edge e and the parameter λ. Every open boundary interval $(\mathbf{x}_1, \mathbf{x}_2)$ contained in e is represented by some *edge interval* $(e, (\lambda_1, \lambda_2))$, $l \leq \lambda_1 \leq \lambda_2 \leq h$. Vertices are represented by themselves.

Definition 6 (Successor Function) *Let e and e' be two edges with $(\mathbf{u}, \mathbf{v})$- and $(\mathbf{u}', \mathbf{v}')$-representations. The partial function $f_{e,e'} : (l, h) \to (l', h')$ is defined as follows: $f_{e,e'}(\lambda) = \lambda'$ iff $\mathbf{x}' = (e', \lambda')$ is the immediate successor of $\mathbf{x} = (e, \lambda)$.*

Claim 4 *Given a representation, the successor function is well-defined and computable.*

Proof: Obviously if there is no P such that $e \subseteq In(P)$ and $e' \subseteq Out(P)$ then $f_{e,e'}$ is nowhere defined. Consider now $(e, \lambda) \in In(P)$ and its successor $(e', \lambda') \in Out(P)$ for some region P having a slope $\mathbf{c}$. For (e', λ') to be indeed the successor of (e, λ), the following vector equation must be satisfied for some $t > 0$:

$$\mathbf{v}' + \lambda'\mathbf{u}' \quad = \quad \mathbf{v} + \lambda\mathbf{u} + t\mathbf{c},$$

by rearranging, we obtain

$$\lambda'\mathbf{u}' \quad = \quad \lambda\mathbf{u} + (\mathbf{v} - \mathbf{v}') + t\mathbf{c}.$$

Multiplying both sides by $\hat{\mathbf{c}}$, the right rotation of $\mathbf{c}$, and observing that $\hat{\mathbf{c}} \cdot \mathbf{c} = 0$, we obtain

$$\lambda'(\hat{\mathbf{c}} \cdot \mathbf{u}') \quad = \quad \lambda(\hat{\mathbf{c}} \cdot \mathbf{u}) + \hat{\mathbf{c}} \cdot (\mathbf{v} - \mathbf{v}')$$

from which we get

$$\lambda' \quad = \quad A_{e,e'}\lambda + B_{e,e'},$$

where

$$A_{e,e'} = \frac{\mathbf{c} \cdot \mathbf{a}}{\mathbf{c} \cdot \mathbf{a}'} \qquad \text{and} \qquad B_{e,e'} = \frac{\hat{\mathbf{c}} \cdot (\mathbf{v} - \mathbf{v}')}{\mathbf{c} \cdot \mathbf{a}'}$$

To obtain these expressions for $A_{e,e'}$ and $B_{e,e'}$, observe that $\hat{\mathbf{c}} \cdot \mathbf{u} = \hat{\mathbf{c}} \cdot \hat{\mathbf{a}} = \mathbf{c} \cdot \mathbf{a}$ and, similarly, $\hat{\mathbf{c}} \cdot \mathbf{u}' = \mathbf{c} \cdot \mathbf{a}'$.

By the assumption of nondegeneracy, $\mathbf{c}$ is not parallel to e' and therefore the denominator is never zero. If λ' is outside the interval (l', h') then the trajectory starting at (e, λ) intersects another border element and $f_{e,e'}(\lambda)$ is undefined. ▪

Similarily we can define a *predecessor* function and compute it by

$$\lambda = \frac{\lambda' - B_{e,e'}}{A_{e,e'}}$$

Claim 5 *If $e \in In(P)$ and $e' \in Out(P)$ then $A_{e,e'} > 0$.*

Proof: Since e is an entry edge of P, $\mathbf{c} \cdot \mathbf{a} > 0$, where $\mathbf{a}$ is the characteristic vector of e which, according to our convention, points into P. As e' is not an entry edge of P, its characteristic vector $\mathbf{a}'$ points away from P, and the normal to e pointing into P is given by $-\mathbf{a}'$. Since $e' \in Out(P)$, $\mathbf{c} \cdot (-\mathbf{a}') < 0$, leading to $\mathbf{c} \cdot \mathbf{a}' > 0$. It follows that $A_{e,e'} = \frac{\mathbf{c} \cdot \mathbf{a}}{\mathbf{c} \cdot \mathbf{a}'} > 0$. ▪

The one-step successor function can be generalized naturally to signatures.

Definition 7 (Signature Successor Function) *Let $\sigma = e_1, \ldots, e_k$ be a signature. The signature successor is a partial function $f_\sigma : (l_1, h_1) \to (l_k, h_k)$ such that $\lambda_k = f_\sigma(\lambda_1)$ iff a trace $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with $\mathbf{x}_1 = (e_1, \lambda_1)$ has the signature σ and $\mathbf{x}_k = (e_k, \lambda_k)$.* We denote the interval on which f_σ is defined by $dom(f_\sigma)$.

It can be easily verified that f_σ can be computed from $f_{e_i, e_{i+1}}$, $i = 1 \ldots k-1$, yielding

$$f_\sigma(\lambda_1) \quad = \quad A_\sigma \lambda_1 + B_\sigma$$

with

$$A_\sigma = A_{e_1,e_2} \cdot A_{e_2,e_3} \ldots A_{e_{k-1},e_k}$$

and

$$B_\sigma = (\cdots((B_{e_1,e_2} \cdot A_{e_2,e_3} + B_{e_2,e_3}) \cdot A_{e_3,e_4} + B_{e_3,e_4}) \cdots) \cdot A_{e_{k-1},e_k} + B_{e_{k-1},e_k}$$

Of particular interest are the successor functions associated with cyclic signatures $\sigma = e_1, \ldots, e_k$ where $e_k = e_1$.

Suppose $\bar{\xi} = \mathbf{x}_1, \mathbf{x}_2, \ldots$ with $\mathbf{x}_i = (e_i, \lambda_i)$ for every $i \in \{1 \ldots k\}$, has a periodic signature $(e_1, \ldots, e_{k-1})^\omega$ and let $\sigma = e_1, \ldots, e_k$. Then the sequence of intersection points of ξ with e_1 is represented by $(e_1, \mu_0), (e_1, \mu_1), \ldots$ with $\mu_0 = \lambda_1$ and $\mu_{i+1} = A_\sigma \cdot \mu_i + B_\sigma$. It is straightforward to solve this linear difference equation and obtain the following expression for μ_n:

$$\mu_n = \begin{cases} \mu_0 + B_\sigma \cdot n & \text{if } A_\sigma = 1 \\ \mu_0 \cdot A_\sigma^n + B_\sigma \cdot \dfrac{A_\sigma^n - 1}{A_\sigma - 1} & \text{otherwise} \end{cases}$$

We can compute the limit of μ_n as n tends to infinity. The possible cases are listed in table 2.

Case	*Limit*
$A_\sigma = 1, B_\sigma = 0$	μ_0
$A_\sigma = 1, \lvert B_\sigma \rvert > 0$	∞
$A_\sigma = 1, \lvert B_\sigma \rvert < 0$	$-\infty$
$A_\sigma < 1$	$\dfrac{B_\sigma}{1 - A_\sigma}$
$A_\sigma > 1, \mu_0 = \frac{B_\sigma}{1-A_\sigma}$	μ_0
$A_\sigma > 1, \mu_0 > \frac{B_\sigma}{1-A_\sigma}$	∞
$A_\sigma > 1, \mu_0 < \frac{B_\sigma}{1-A_\sigma}$	$-\infty$

Table 2: *The limits of the intersection sequence* $\mu_{n+1} = A_\sigma \mu_n + B_\sigma$ *as a function of* A_σ, B_σ *and* μ_0.

For every e_i, $i \in \{1, \ldots, k-1\}$ in the signature we define its limit point λ_i^* by letting $\lambda_1^* = \mu^*$ and $\lambda_i^* = f_{e_{i-1}, e_i}(\lambda_{i-1}^*)$. Clearly the signature $e_1, \ldots, e_{k-1}$ can repeat forever only if for every i, $i \in \{1, \ldots, k-1\}$ all its intersection points with the edge e_i are contained in the interval $(e_i, (l_i, h_i))$. We can now formulate a criterion for a cycle being repeated infinitely many times (see figure 9).

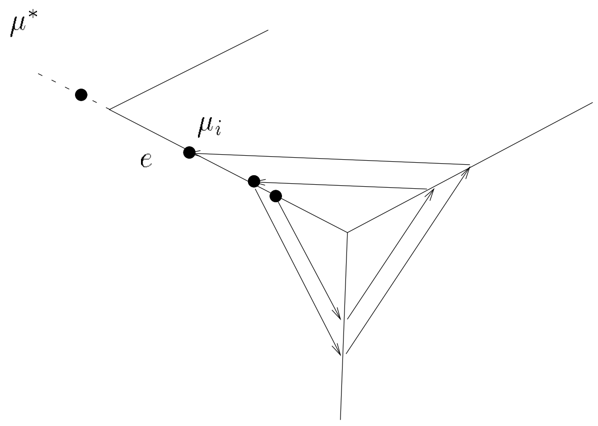


Figure 9: *A non-ultimate spiral: the limit of the sequence of intersection points with e is beyond the endpoint.*

Criterion 1 (Infinity test) *A trajectory starting with the cycle $(e_1, \lambda_1), \ldots, (e_k, \lambda_k)$ such that $e_1 = e_k$, has a periodic signature $(e_1, \ldots, e_{k-1})^\omega$ iff*

$$l_i \leq \lambda_i^* \leq h_i \qquad \textit{for every } i \in \{1, \ldots, k-1\}$$

This criterion is based on the first occurrence of the cycle and enables us to decide in advance whether this cycle will repeat forever or some of its edges will be abandoned after finitely many iterations.

Theorem 6 (Point–to–Point Reachability) *The problem* $\mathbf{Reach}(\mathcal{H}, \mathbf{x}, \mathbf{x}')$ *is decidable.*

Proof: We first show that it is decidable when $\mathbf{x}$ and $\mathbf{x}'$ are boundary points, $\mathbf{x} = (e, \lambda)$ and $\mathbf{x}' = (e', \lambda')$. All we need is to calculate the successors and compare them with $\mathbf{x}'$. During the generation of the trajectory we keep track of the non-abandoned edges (finitely many). When a cycle is detected the infinity test tells us whether or not this is the terminal cycle. If it is not we continue generating the successors. If it is the terminal cycle we can check whether e' is not in the cyclic signature or λ' is beyond the limit – in that case the answer is negative. Otherwise, after finitely many iterations we either reach $\mathbf{x}'$ or bypass it. This also implies that the problem is solvable for arbitrary points, because it is straightforward to compute the forward and backward intersections of trajectories passing through an interior point of a region with the boundaries. ∎

If the target λ' is *exactly* the limit of the sequence, the reachability problem becomes an instantiation of Zeno's paradox, and the answer is a matter of definition.

5 Reachability Between Edges

The results so far allow us to compute the set of all points reachable by a single trajectory. Now we will show how reasoning about a non-countable number of trajectories departing from an edge interval can be reduced to reasoning about a finite number of trajectories.

Successor Trees

Claim 7 *Let (e', l') and (e', h') be the successors of (e, l) and (e, h) respectively. Then for every λ, $l < \lambda < h$ the successor of (e, λ) is some (e', λ'), $l' < \lambda' < h'$.*

Proof: Follows from the monotonicity of the successor function and the convexity of the regions. ∎

Consequently, for any edge interval $(e, (l, h))$, if the trajectories starting from (e, l) and (e, h) have identical signatures, so does every trajectory starting at (e, λ), for some $l < \lambda < h$.

Next, we generalize the notion of a successor from single points to intervals.

Definition 8 (Edge Successors Set) *Let $(e, (l, h)) \subseteq In(P)$ be an edge interval. The successor set of e is a set of edges and vertices*

$$Succ(e, (l, h)) = \{(e_1, (l_1, h_1)), \mathbf{x}_1, (e_2, (l_2, h_2)), \mathbf{x}_2, \ldots, (e_n, (l_n, h_n))\}$$

such that the elements are mutually disjoint, and their union is a connected boundary interval being in one-to-one correspondence with $(e, (l, h))$ via the successor/predecessor functions. (See figure 10).

Claim 8 *The set of edge-successors is finite and computable.*

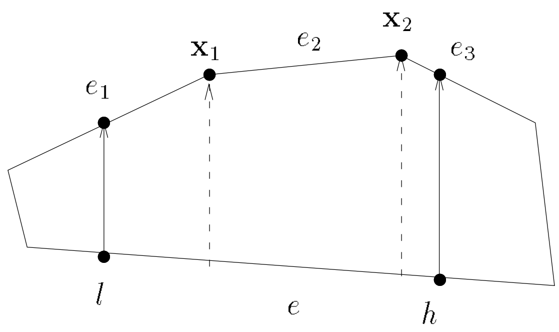


Figure 10: *The successors of* $(e,(l,h))$.

Proof: Because of the computability of the single point successor and predecessor functions. Moreover, for every i, all the points in $(e_i,(l_i,h_i))$ are the f_{e,e_i}-images of some sub-interval of $(e,(l,h))$. ∎

Clearly the set of all points in $Out(P)$ which are immediate successors of points in $(e,(l,u))$ is exactly the union of all elements in $Succ(e,(l,u))$.

Definition 9 (Successor Tree) *A successor tree rooted at* $(e,(l,h))$ *is constructed by calculating recursively the successors of every node starting at the root* (see figure 11). *A path along the tree is called a successor chain.*[2]

The notion of signature is generalized naturally to successor chains. If a chain has a signature σ then there must be at least one trajectory having this signature. Hence every infinite successor chain must have an ultimately periodic signature.

The idea behind the edge-to-edge reachability algorithm is that all trajectories starting in some edge-interval $(e,(l,h))$ and having the same signature are characterized by the behavior of two trajectories: the "leftmost" trajectory starting at l and the "rightmost" trajectory starting at h. By this we mean that

1. For trajectories with a finite signature $\sigma = e,\ldots,e'$, the existence of a trajectory from some $\mathbf{x} \in (e,(l,h))$ to some $\mathbf{x}' \in (e',(l',h'))$ is exactly

[2] For the sake of simplicity we ignore here the simpler case where one of the successors is a vertex – in this case a successor chain is simply a trace.

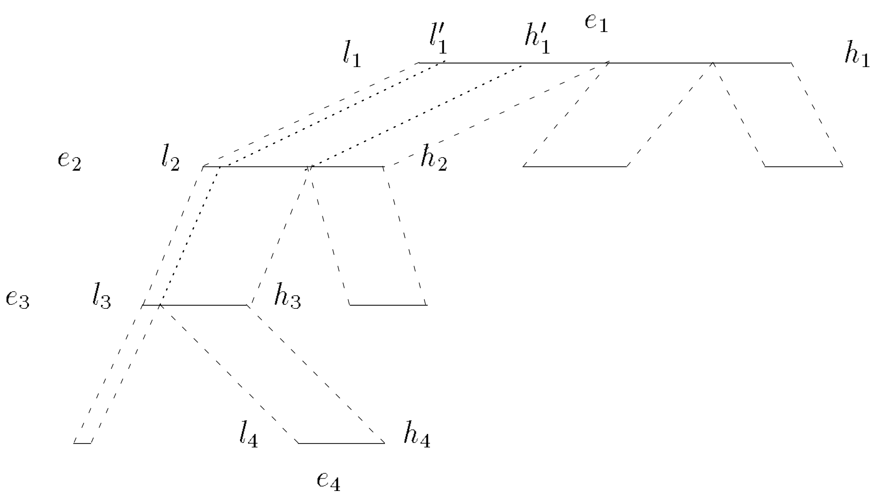


Figure 11: *A finite portion of a successor-tree with a chain* $(e_1,(l_1,h_1)),(e_2,(l_2,h_2)),(e_3,(l_3,h_3)),(e_4,(l_4,h_4))$. *All the trajectories starting at some point in* $(e_1,(l'_1,u'_1))$ *have the same signature* e_1,e_2,e_3,e_4.

the existence of a non-empty intersection between the intervals (l',h') and $(f_\sigma(l),f_\sigma(h))$

2. The criterion for a given cyclic signature to repeat forever is based on the convergence of the sequences associated with l and h.

For an edge interval $(e,(l,h))$ and a cyclic signature $\sigma = e,\ldots,e$, we define

$$f_\sigma((l,h)) = \{\lambda' : \lambda' = f_\sigma(\lambda) \text{ for some } \lambda \in (l,h)\}$$

Claim 9 (Decidability for Periodic Chains) *Consider the cyclic successor-chain* $(e_1,(l_1,h_1)),\ldots,(e_k,(l_k,h_k))$, $(e_1 = e_k)$, *and let* $\sigma = e_1,\ldots e_{k-1}$. *It is decidable whether this finite successor-chain can be extended to an infinite periodic chain with the signature* σ^ω, *and if it is periodically extensible, whether there exists a trajectory among all those which share the same chain, reaching a point in an edge-interval* $(e',(l',h'))$.

Proof: Consider the sequence of intervals $(\mu_0,\eta_0),(\mu_1,\eta_1),\ldots$ defined by $(\mu_0,\eta_0) = (l_k,h_k)$ and $(\mu_{n+1},\eta_{n+1}) = (A_\sigma\mu_n + B_\sigma, A_\sigma\eta_n + B_\sigma)$. In table 3, we present the different cases that may arise. For each of them, we specify the conditions for infinite periodic extension of the cyclic successor chain and the set of points on edge e_1 which are reachable by such an extension if it exists. If no infinite extension exists, the third column is irrelevant. In this table, λ^* denotes $B_\sigma/(1-A_\sigma)$, and we assume that e_1 is represented by the interval (L, H).

Case	Extensibility Condition	e_1-reachability set
$a1.\quad A_\sigma = 1, B_\sigma = 0$	Always	(μ_0, η_0)
$a2.\quad A_\sigma = 1, B_\sigma > 0$	$(\mu_0, \infty) \subseteq dom(f_\sigma)$	$\bigcup_{n\geq 0}(\mu_n, \eta_n)$
$b1.\quad A_\sigma < 1, \lambda^* = \eta_0$	Always	(μ_0, η_0)
$b2.\quad A_\sigma < 1, \eta_0 < \lambda^*$	$\lambda^* \in dom(f_\sigma)$	$\bigcup_{n\geq 0}(\mu_n, \eta_n)$
$c1.\quad A_\sigma > 1, \lambda^* = \mu_0$	Always	$f_\sigma((\mu_0, H))$
$c2.\quad A_\sigma > 1, \lambda^* < \mu_0$	$(\mu_0, \infty) \subseteq dom(f_\sigma)$	$\bigcup_{n\geq 0}(\mu_n, \eta_n)$

Table 3: Extensibility conditions and reachability sets.

Note that the condition $(\mu_0, \infty) \subseteq dom(f_\sigma)$ implies that all the edges in σ are unbounded from the right.

The cases

$A_\sigma = 1, B_\sigma < 0 \qquad A_\sigma < 1, \lambda^* < \mu_0 \qquad A_\sigma > 1, \eta_0 < \lambda^*$

are treated symmetrically to the cases

$A_\sigma = 1, B_\sigma > 0 \qquad A_\sigma < 1, \eta_0 < \lambda^* \qquad A_\sigma > 1, \lambda^* < \mu_0$

The cases $A_\sigma \neq 1, \mu_0 < \lambda^* < \eta_0$ are treated by splitting the cyclic successor chain

$$(e_1, (l_1, h_1)), \ldots (e_k, (l_k, h_k))$$

into the two chains

$$(e_1, (l_1, \lambda^*)), \ldots (e_k, (l_k, \lambda^*)) \qquad \text{and} \qquad (e_1, (\lambda^*, h_1)), \ldots (e_k, (\lambda^*, h_k))$$

and considering each of them separately.

Without loss of generality, we may assume that $e' = e_1$, i.e., that we wish to check for the reachability of an interval which lies on the initial edge e_1. To check that some point in $(e', (l', h')$ is reachable, we have to check whether (l', h') intersects the interval (μ_0, η_0) (cases $a1$ and $b1$) or the interval $f_\sigma((\mu_0, H))$ (case $c1$). For cases $a2$, $b2$, and $c2$, it is necessary to check whether (l', h') intersects any of the intervals (μ_n, η_n), for some $n \geq 0$. Since, in all three cases, the two sequences $\mu_0, \mu_1, \ldots$ and $\eta_0, \eta_1, \ldots$ are monotonically increasing, and have closed form expressions, it is straightforward to check for the necessary intersection. ∎

Definition 10 (Non-redundant Successor Tree) *A non-redundant successor tree is obtained from a successor tree via the following recursive transformation: if $(e, (l', h'))$ is a cyclic successor of $(e, (l, h))$ then replace it by $(e, (l', h')) - (e, (l, h))$.*

Usually, the modified successor will be a single interval, but in the case of $A_\sigma > 1$ and $l < \frac{B_\sigma}{1-A_\sigma} < h$ the result might be two non-connected intervals of the same edge – see figure 5. Clearly, in terms of *reachable states* the non-redundant tree carries the same information as the original tree, namely the union of the edges is still the set of reachable boundary points.

A node in a tree is called a *branching node* if it has more than one successor.

Claim 10 (Finitely Many Chains) *Every infinite chain in a non-redundant successor tree, has only finitely many branching nodes. Hence this tree contains only finitely many chains.*

Proof: Since all infinite chains are ultimately periodic it suffices to consider the periodic part of such chains. Let $(e_1, (l_1, h_1), (e_2, (l_2, h_2)), \ldots$ be an infinite successor chain with a periodic signature σ^ω, $\sigma = (e_1, \ldots, e_{k-1})$. Let $(\mu_0, \eta_0), (\mu_1, \eta_1) \ldots$, denote the intervals in the successor chain that correspond to successive visits at the edge e_1, starting at the second. Thus, $(\mu_0, \eta_0) = (l_k, h_k)$, $(\mu_1, \eta_1) = (l_{2k}, h_{2k})$, etc.

We consider each of the six cases of table 3 and show that each of them contains only finitely many branching nodes, under the infinite extensibility condition. Note that the chains we consider here obey the non-redundance

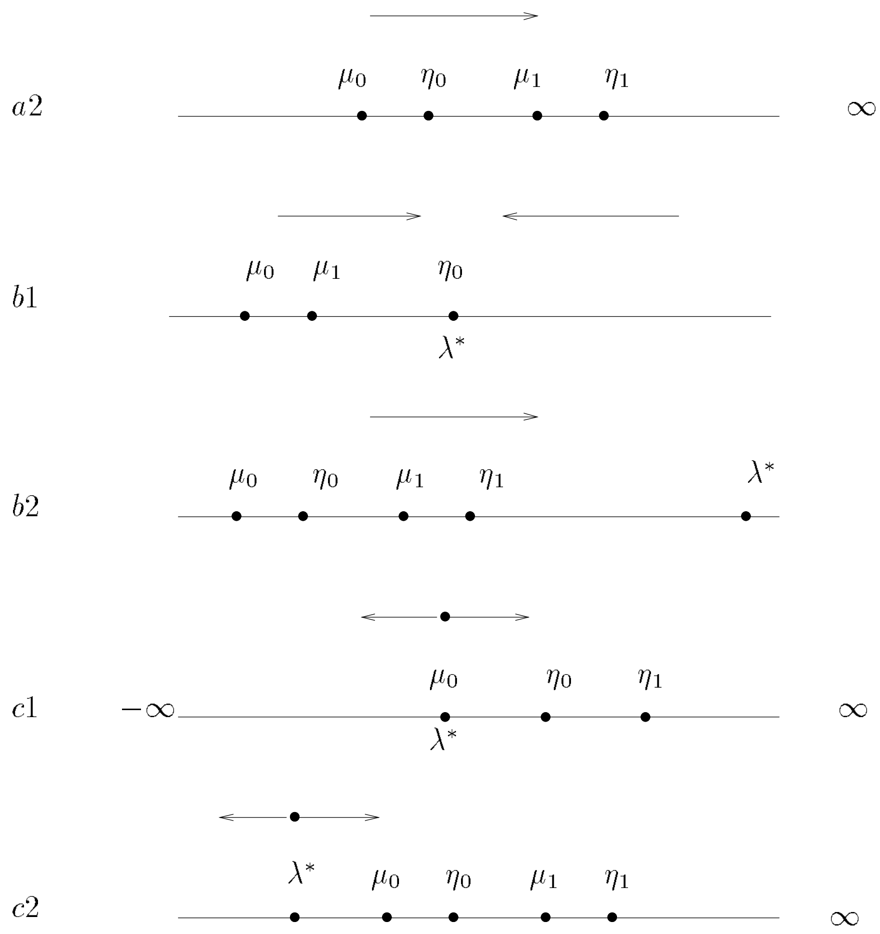


Figure 12: *An illustration of cases a2–c2 of table 3*

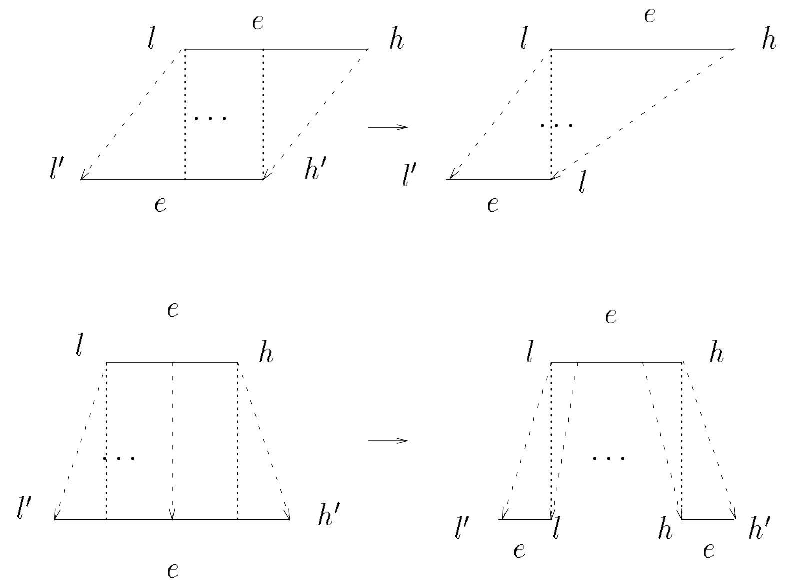


Figure 13: *Transforming a successor-tree (left) into a non-redundant one (right): the non-split case (up) and the split case (down).*

condition. In particular, the intervals (μ_i, η_i) and (μ_j, η_j) are disjoint for every $i \neq j$. Also note that branching can arise between the nodes $(e_1, (\mu_i, \eta_i))$ and $(e_1, (\mu_{i+1}, \eta_{i+1}))$ only if there exists some $\lambda \in (\mu_i, \eta_i)$ such that $f_\sigma(\lambda)$ is undefined.

We consider several cases together.

- Cases $a1 : A_\sigma = 1, B_\sigma = 0$ and $b1 : A_\sigma < 1, \lambda^* = \eta_0$.
  In these two cases, $f_\sigma(\lambda)$ is defined for every $\lambda \in (\mu_0, \eta_0)$ and $f_\sigma((\mu_0, \eta_0)) \subseteq (\mu_0, \eta_0)$. Consequently, all the e_1-points reachable after one round of σ are redundant and the non-redundant chain cannot be infinite.

- Case $a2 : A_\sigma = 1, B_\sigma > 0$, case $b2 : A_\sigma < 1, \eta_0 < \lambda^*$, and case $c2 : A_\sigma > 1, \lambda^* < \mu_0$.
  It can be shown that, under the infinite extensibility condition, $f_\sigma(\lambda)$ is defined for every $\lambda \in (\mu_k, \eta_k)$ and $(\mu_{k+1}, \eta_{k+1}) \subseteq f_\sigma((\mu_k, \eta_k))$, for every $k \geq 0$. It follows that this chain cannot have a branching node following the second visit to e_1.

- Case $c1: A_\sigma > 1, \lambda^* = \mu_0$.
  As $\mu_0 < \eta_0$, there is a sufficiently small $\epsilon > 0$ such that $f_\sigma((\mu_0, \mu_0 + \epsilon)) \subseteq (\mu_0, \eta_0)$. By non-redundancy, this implies that $\lambda^* < \mu_1$ which shows that from the second σ-round on, this case behaves like case $c2: A_\sigma > 1, \lambda^* < \mu_0$.

It follows that every chain in the non-redundant tree has finitely many branching points. By a König-like argument, this implies that the tree has only finitely many chains, some of which may still be infinite. ∎

Consequently we have:

Theorem 11 (Edge-to-edge Reachability) *For every deterministic planar PCD system it is decidable whether an edge interval* $(e'(l', h'))$ *is reachable from an edge interval* $(e, (l, h))$.

Proof: The algorithm develops top-down the non-redundant successor tree starting at $(e, (l, h))$ where along every chain it keeps track of unabandoned ancestors. Using this information we can detect edge-cycles and calculate their successor functions and limits. All the infinite chains are ultimately periodic and there are only finitely many of them. Once, it is realized, using the criteria of Claims 9 and 10, that the currently examined chain has a non-branching infinite periodic non-redundant continuation, we need not expand it any longer, but can use the methods of Claim 9 to decide whether it ever intersects $(e', (l', h'))$. ∎

As an immediate result we have:

Corollary 12 (Region to Region Reachability) *Let R and R' be two finite unions of polygonal sets. Then the problem* $\mathbf{Reach}(\mathcal{H}, R, R')$ *is decidable.*

Proof: Inside the regions we have two-sided determinism so every region-to-region reachability problem can be reduced to a finite number of edge-to-edge reachability problems. ∎

6 Undecidability for Three Dimensions

The particular topological properties of the plane played a major role in the convergence of our decision procedure. In this section we show that one additional dimension renders the reachability problem undecidable. This negative result will be based on demonstrating the computational power of PCD systems. We will show that 3-dimensional PCD systems can simulate, in a sense described below, arbitrary Turing machines.

A *deterministic transition system* is $\mathcal{A} = (Q, \delta)$ where Q is a countable set of states and $\delta : Q \to Q$ is the transition function. A *run* of $\mathcal{A}$ starting at some q is a (finite or infinite) sequence $\tau = q[0], q[1], \ldots$ where $q[0] = q$ and $q[i] = \delta(q[i-1])$ for every $i > 0$. The set of all such sequences(*runs*) starting at some $q \in Q_0 \subseteq Q$ is denoted by $L(\mathcal{A}, Q_0)$. We use $\delta^i(q)$ to denote $\delta(\delta^{i-1}(q))$ with $\delta^1(q) = \delta(q)$.

Definition 11 (Simulation) *Let $\mathcal{A} = (Q, \delta)$ be a deterministic transition system and let Q_0 be a subset of Q. We say that $\mathcal{A}$ is Q_0-simulated by a PCD system $\mathcal{H} = (X, f)$ if there exists a subset Y of X and a bijection $\pi : Y \to Q_0$ (state-mapping) such that for every $\mathbf{x}, \mathbf{x}' \in Y$, there is a trajectory in $\mathcal{H}$ from $\mathbf{x}$ to $\mathbf{x}'$ (not intersecting Y except in $\mathbf{x}$ and $\mathbf{x}'$) iff for some i $\delta^i(\pi(\mathbf{x})) = \pi(\mathbf{x}')$ and for every j, $0 < j < i$, $\delta^j(\pi(\mathbf{x})) \notin Q_0$.*

By transitivity a trajectory between any $\mathbf{x}$ and $\mathbf{x}'$ in Y exists iff there is a run $\tau = q[0], \ldots q[m] \in L(\mathcal{A}, Q_0)$ such that $\pi(\mathbf{x}) = q[0]$ and $\pi(\mathbf{x}') = q[m]$. When $Q_0 = Q$ we say simply that $\mathcal{H}$ simulates $\mathcal{A}$ and in this case an algorithm for solving the reachability problem for $\mathcal{H}$ solves the reachability (and in particular, the halting) problem for $\mathcal{A}$.

Remark: There is a variety of other notions of simulation between discrete and continuous dynamical systems, but these semantical issues are subject of an independent ongoing research (see [2], [5]). The simple definition we use here is sufficient for the purpose of proving undecidability.

6.1 Simulation of Finite-State Machines

Here we show how every finite-state automaton can be simulated by a 3-dimensional PCD system. A finite automaton without input is a rather trivial object and the construction is presented here just because it underlies the more complicated constructions for infinite-state machines.

Claim 13 (Simulation of Finite Automata) *Every finite deterministic automaton can be simulated by a 3-dimensional PCD system.*

Proof: Suppose the automaton has n states. The simulating system is defined over the subset $[1,n]\times[0,1]\times[0,n]$ of $\mathbb{R}^3$. It consists of the following regions (we call the state variables x, y, and z):

Region	Defining conditions	$\mathbf{c}=(\dot{x},\dot{y},\dot{z})$
F	$(z=0)\wedge(y<1)$	$(0,1,0)$
U_{ij}	$(x=i)\wedge(y=1)\wedge(z<j)$	$(0,0,1)$
B_{ij}	$(z=j)\wedge(x+(j-i)y=j)\wedge(y>0)$	$(j-i,-1,0)$
D	$(z>0)\wedge(y=0)$	$(0,0,-1)$

The regions U_{ij} and B_{ij} are defined for every i,j such that $\delta(q_i)=q_j$. As a state-mapping we take $\pi(x,y,z)=q_i$ iff $(x=i)\wedge(y=z=0)$. An example of this construction appears in figure 14. It can be verified that the system goes from $(i,0,0)$ to $(j,0,0)$ iff $\delta(q_i)=q_j$. At F the system advances y until $(i,1,0)$, then at U_{ij} it goes "up" until it reaches the surface $z=j$ at $(i,1,j)$. On that surface in region B_{ij} it goes diagonally to $(j,0,j)$ and finally in D it goes down to $(j,0,0)$. Note that all the regions leading to $(j,0,j)$ are located on the same plane. ∎

Remark: This technique can be applied to the simulation of non-deterministic automata by *non-deterministic* PCD systems. All we have to do is to modify the definition of a PCD system to allow non-determinism on the (relative) boundaries of the regions. Then if both (q_i,q_j) and (q_i,q_k) are possible transitions, $j<k$, the system will bifurcate in $(i,1,j)$ between B_{ij} (going to q_j) and U_{ik} (going up until $z=k$ and then to B_{ik}). In [2] we have shown that deterministic PCD systems can simulate (in a richer sense) non-deterministic automata.

6.2 Push-Down Automata

The results concerning infinite-state transition-systems will be based on stack machines. A pushdown stack is an element of Σ^*0^ω where[3] $\Sigma=\{0,\ldots,k-1\}$. We define the following two functions: $\textsc{push}$: $\Sigma\times\Sigma^\omega\to\Sigma^\omega$ and $\textsc{pop}$: $\Sigma^\omega\to\Sigma\times\Sigma^\omega$ as $\textsc{push}(v,S)=v\cdot S$ and $\textsc{pop}(v\cdot S)=(v,S)$.

Definition 12 (PDAs) *A deterministic pushdown-automaton (PDA) is a transition system $\mathcal{A}=(Q\times\Sigma^*0^\omega,\delta)$ for some $Q=\{q_1,\ldots q_n\}$ such that δ*

[3] It is more convenient to define the set of all stacks as a countable subset of Σ^ω although it is isomorphic to Σ^*.

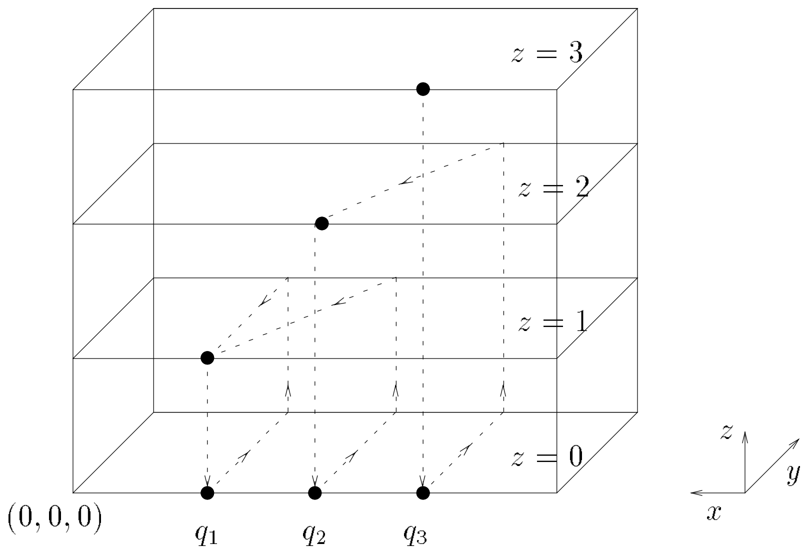


Figure 14: Simulating a 3-state automaton with $\delta(q_1) = \delta(q_2) = q_1$ and $\delta(q_3) = q_2$.

is defined using a finite collection of statements of one of the following two forms:

q_i: $S :=$ PUSH(v, S);	q_i: $(v, S) :=$ POP(S);
GOTO q_j	IF $v = 0$ GOTO q_{i_0};
	...
	IF $v = k - 1$ GOTO $q_{i_{k-1}}$;

The contents of a stack is denoted by $S = s_1 s_2 \dots$ where s_1 is the top of the stack. We define the standard encoding function $r : \Sigma^\omega \to [0, 1]$ as $r(S) = \sum_{i=1}^{\infty} s_i k^{-i}$. Clearly r is injective on $\Sigma^* 0^\omega$. It is easily verified that the stack operations have arithmetic counterparts that operate on the representation:

$$\begin{array}{rl} S' = \text{PUSH}(v, S) & \text{iff} \quad r(S') = (r(S) + v)/k \\ (S', v) = \text{POP}(S) & \text{iff} \quad r(S') = kr(S) - v \end{array}$$

Claim 14 (Simulation of PDAs) *Every PDA can be simulated by a 3-dimensional PCD system.*

Proof: For simplicity we assume $k = 2$ and $\Sigma = \{0, 1\}$. Consider the three planar sub-systems depicted in figure 15 and a trajectory segment starting at $\mathbf{x} = (x, 0)$, $x \in [0, 1]$ and ending at $\mathbf{x}' = (x', 1)$. It can be verified that either:

$$\begin{array}{ll} x' = (x+1)/2 & \textsc{push } 1 \\ x' = x/2 & \textsc{push } 0 \\ x' = 2x - 1/2 & \textsc{pop} \end{array}$$

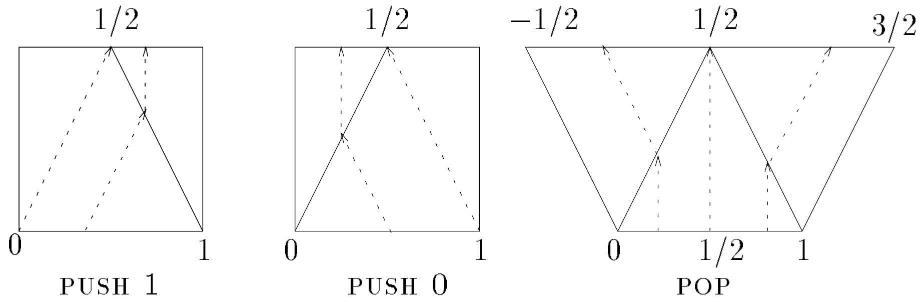


Figure 15: The basic elements.

If $x = r(S)$ at the "input port" ($y = 0$) of a PUSH element, then $x' = r(S')$ at the "output port" ($y = 1$) of that element where S' is the resulting stack. For the POP element we have two output ports $-1/2 \leq x < 1/2$ and $1/2 \leq x < 3/2$. If the top of the stack was 0 the trajectory reaches the left port with $x' = r(S') - 1/2$, otherwise it goes to the right port with $x' = r(S') + 1/2$. In both cases the value of x' (relative to the "origin" of the port) encodes the new content of the stack. Thus, in order to simulate a PDA we pick for every q_i an element corresponding to its stack operation, place it with the origin in position, say, $(2i, 0, 0)$ and use the third dimension in order to connect the output ports back to the input ports according to the GOTO's (see figure 16). This is similar to the previous construction except for the fact that the connections are via two-dimensional "bands" and thus two families of trajectories going to the same state q_i cannot be merged on the same plane ($z = j$) but only while going "down". Finally the state-mapping is defined as $\pi(x, y, z) = (q_i, S)$ iff $y = z = 0$, $2i \leq x < 2i + 1$ and $S = r^{-1}(x - 2i)$. ∎

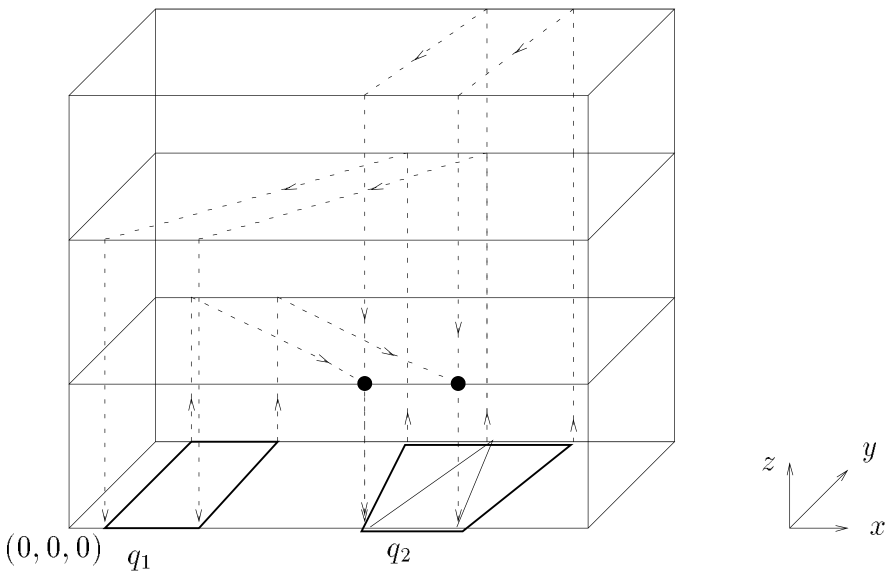


Figure 16: Simulating a PDA with 2 states, defined by: $q_1 : S :=$PUSH$(1, S)$; GOTO q_2; $q_2 : (v, S) :=$POP(S); If $v = 1$ THEN GOTO q_2 ELSE GOTO q_1. Note the place where the two GOTOs to q_1 merge.

6.3 Simulating Two-stack and Turing Machines

The construction of claim 14 generalizes naturally to automata having two stacks (2PDAs). We can define an encoding function $\bar{r} : \Sigma^\omega \times \Sigma^\omega \rightarrow [0,1] \times [0,1]$ by letting $\bar{r}(S_1, S_2) = (r(S_1), r(S_2))$. This way every configuration of the two stacks can be encoded by a point $\mathbf{x} = (x_1, x_2, 0)$ in a two-dimensional input port. The elements that simulate the stack operations PUSH(v, S_1), PUSH(v, S_2), POP(S_1) and POP(S_2) operate on the appropriate dimension, according to the stack involved, and leave the other dimension intact. As an example, an element corresponding to PUSH$(0, S_1)$ appears in figure 17. From this we can immediately conclude:

Claim 15 *Every 2PDA (and hence any Turing machine) can be simulated by a 4-dimensional PCD system.*

Proof: As in claim 14 we pick n elements, arrange them along a line and connect output ports to input ports. The connections should now "carry"

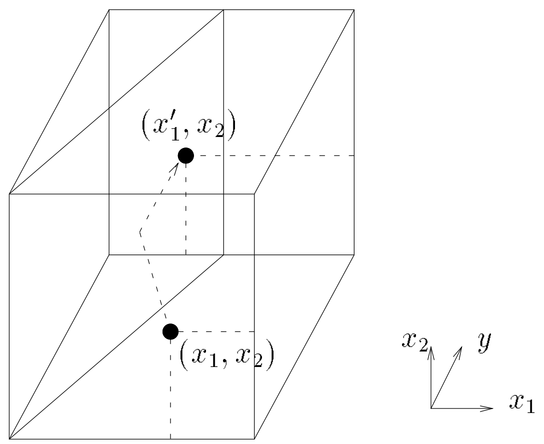


Figure 17: An element simulating the operation PUSH$(0, S_1)$

two-dimensional information about the configuration, and thus consist of three-dimensional "tubes". Several tubes going to the same state can be merged by employing a fourth dimension (no figure) in the same way as two-dimensional "bands" were merged in the case of one-stack PDAs. ∎

But we can do better. First, recall from the standard proof of the equivalence of Turing machines and 2PDAs that some constraints can be imposed on the type of 2PDAs used.

Definition 13 (Normal 2PDA) *Let $\Sigma = \{0, 1, 2\}$. A configuration $(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega$ is normal if both S_1 and S_2 belong to $\{1, 2\}^* 0^\omega$. A 2PDA is normal if it never pushes 0 to any of the stacks.*

It can be easily verified that normality of the configurations is preserved by normal 2PDAs and that normal 2PDAs can simulate Turing machines (see the proof of equivalence of Turing machines and 2PDAs, e.g., [11]).

Definition 14 *Let $\mathcal{A} = (Q \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta)$ be a 2PDA and let C_0 be a set of configurations. With every $q_i, q_j \in Q$ we associate the set*

$$\begin{aligned} S(C_0, i, j) \;=\; & \{(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega : (\exists \tau \in L(\mathcal{A}, C_0)(\exists k > 0) \\ & \tau[k] = (q_i, S_1', S_2') \wedge \tau[k+1] = (q_j, S_1, S_2)\} \end{aligned}$$

In other words, $S(C_0, i, j)$ is the set of all 2-stack configurations with which a transition from q_i to q_j can take place in any run of $\mathcal{A}$ starting at C_0.

Definition 15 (Separated and Flat States, Regular 2PDAs) *A state q_k of a 2PDA is separated if for every $q_i \neq q_j \in Q$, if both q_i and q_j have a* GOTO q_k *instructions, they are preceded respectively by* PUSH v_i *and* PUSH v_j *(into the same stack) for $v_i \neq v_j$. A state q_k of a 2PDA is C_0-flat if for every j, $S(C_0, j, k) \subseteq \{0^\omega\} \times \Sigma^\omega$. A 2PDA is C_0-regular if each of its states is either separated or C_0-flat.*

A state q_k is thus separated if, upon entering q_k, the values on the tops of the stacks are sufficient to tell whether we come from q_i or q_j. A state q_k is C_0-flat if in all the runs starting at C_0 it is always entered with the first stack empty.

Claim 16 (Simulation of Regular 2PDAs) *Let C_0 be a set of configurations. Any C_0-regular 2PDA can be C_0-simulated by a* 3*-dimensional PCD system.*

Proof: In the proof of claim 15 we needed the fourth dimension only in order to merge two or more "tubes" entering the same input port associated with a state q. If q is separated these tubes do not overlap on the input port and the connections can be made. If q is flat, the relevant information at the input port of q is one-dimensional and all incoming "bands" can be glued together (see figure 18). ∎

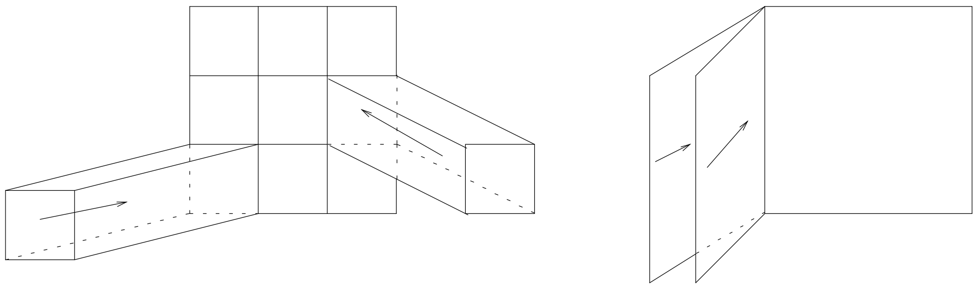


Figure 18: Realizing entrances to input ports of separated (left) and flat (right) states.

Note that the relativity of flatness and of simulation with respect to C_0 is important. A trajectory starting at some configuration outside C_0 might want to enter the input port of q_k at some point with $x_1 \neq 0$, but since we are only interested in C_0-simulation we do not care.

What we are going to show is, informally speaking, that every normal 2PDA can be transformed into an N-regular 2PDA where N is the set of normal configurations. The idea is simple: each time after performing a stack operation, we empty one stack while pushing its contents into the other. Then we perform the GOTO's, that is, merge several "bands" that contain one-dimensional information. Before entering the new input port we decode the one-dimensional representation back into two stacks.

For every $i, j \in \{1, \dots, n\}$, we define a machine $Encoder_{ij}$ and a machine $Decoder_j$. An encoder takes two normal stack configurations $S_1 = \alpha_1, \dots \alpha_l 0^\omega$ and $S_2 = \beta_1, \dots \beta_m 0^\omega$ ($\alpha_i, \beta_i \in \{1, 2\}$) and converts them into $S_1 = 0^\omega$ ("empty" stack) and $S_2 = \alpha_l \dots \alpha_1 0 \beta_1 \dots \beta_m 0^\omega$. The decoder does the reverse operation. These two machines are described below:

Encoder$_{ij}$

E–Entry$_{ij}$:	$S_2 :=$PUSH$(0, S_2)$
	GOTO E–Loop$_{ij}$
E–Loop$_{ij}$:	$(S_1, v) :=$POP(S_1)
	IF $v = 0$ GOTO E–Exit$_{ij}$
	IF $v = 1$ GOTO E–Mov1$_{ij}$
	IF $v = 2$ GOTO E–Mov2$_{ij}$
E–Mov1$_{ij}$:	PUSH$(1, S_2)$
	GOTO E–Loop$_{ij}$
E–Mov2$_{ij}$:	PUSH$(2, S_2)$
	GOTO E–Loop$_{ij}$
E–Exit$_{ij}$:	GOTO D–Entry$_j$

Decoder$_j$

D–Entry$_j$:	$S_1 :=$PUSH$(0, S_1)$
	GOTO D–Loop$_j$
D–Loop$_j$:	$(S_2, v) :=$POP(S_2)
	IF $v = 0$ GOTO D–Exit$_j$
	IF $v = 1$ GOTO D–Mov1$_j$
	IF $v = 2$ GOTO D–Mov2$_j$
D–Mov1$_j$:	PUSH$(1, S_1)$
	GOTO D–Loop$_j$
D–Mov2$_j$:	PUSH$(2, S_1)$
	GOTO D–Loop$_j$
D–Exit$_j$:	GOTO q_j

Claim 17 (Normal $\Rightarrow$ N-Regular) *Let $\mathcal{A} = (Q \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta)$ be a normal 2PDA and let N be the set of all normal configurations. Then there is an N-regular 2PDA $\mathcal{A}' = ((Q \cup Q') \times \Sigma^* 0^\omega \times \Sigma^* 0^\omega, \delta')$ such that for every $q, q' \in Q$ and every $(S_1, S_2), (S_1', S_2') \in N$ there is a run from (q, S_1, S_2) to (q', S_1', S_2') in $\mathcal{A}$ iff there is such a run in $\mathcal{A}'$.*

Proof: We let Q' be the union of the sets of states of the corresponding encoders and decoders. The transition function δ' is the union of the transitions of the encoders and decoders and the following variation of δ: every original $\mathcal{A}$-statement of the form q_i: ... GOTO q_j is replaced by q_i: ... GOTO E–Entry$_{ij}$. It can be easily verified that every q_j is now separated (it is entered only from D–Exit$_j$), that E–Entry$_{ij}$ is separated (it is entered only from q_i) and that D–Entry$_j$ is N-flat (all trajectories starting with a normal configuration will enter the encoder with a normal configuration and will leave the encoder with one stack empty). The other states of the decoders and encoders are obviously separated. This construction is drawn schematically in figure 19. ∎

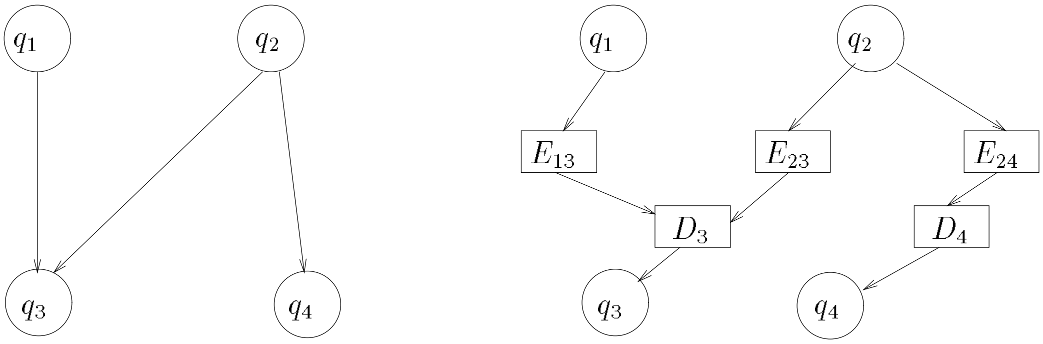


Figure 19: Augmenting a 2PDA with encoders and decoders.

Theorem 18 (Simulation of 2PDAs) *Any normal 2PDA $\mathcal{A}$ can be simulated by a 3-dimensional PCD system.*

Proof: We convert $\mathcal{A}$ into the N-regular 2PDA $\mathcal{A}'$ described above. This 2PDA can be N-simulated in 3 dimensions. By letting $\pi(x, y, z) = (q_i, S_1, S_2)$ iff $y = 0$, $2i \leq x < 2i + 1$, $S_1 = r^{-1}(x - 2i)$, $S_2 = r^{-1}(z)$ and both S_1 and S_2 are normal, we obtain the desired simulation.

Corollary 19 (Undecidability) *The reachability problem for 3-dimensional PCD systems is undecidable.*

Proof: Otherwise we could translate every reachability problem of a Turing machine, into a reachability problem between two rational points in a 3-dimensional simulating system, and solve the halting problem. ∎

7 Discussion

We have defined PCD systems, a class of simple dynamical systems, which is a natural extension of real-time systems (such as timed automata or integration graphs) whose trajectories can be computed effectively and precisely. The class of behaviors exhibited by PCD systems is rather rich even without exploiting the full power of the HS model. We have utilized the constraints on behavior imposed by the topological structure of the plane in order to devise a decision procedure for the reachability problem in the two-dimensional case. We have shown that in higher dimensions, when these properties do not hold anymore, the problem becomes undecidable.

The systems constructed for the undecidability results do not seem to be "naturally occurring" and an open question that remains is what additional restrictions on a PCD system will make the reachability problem decidable regardless of dimensionality. Even for the undecidable cases, our techniques can serve as a basis for semi-decidable symbolic simulation techniques as advocated in [16]. The idea is to start with a formula (= a region of possible initial conditions) and calculate successively formulas that characterize reachable states. Then, if the process converges, a test for intersection between the region expressed by the obtained formula and the target region should be performed. Our decidability result can be viewed as an "on-the-fly" version of the approach suggested in [16], and it works even in the case where no finite formula over the reals can characterize the reachable states (as in the case of a spiral).

The applicability of these methods to more general hybrid systems, having non-constant derivatives in every state, is currently under investigation. This general case poses a new dimension of problems associated with the inability to calculate trajectories and solve equations using exact methods and hence the need for numerical approximations.

In addition to the undecidability result we have shown (in [2]) additional interesting connections between topological properties of dynamical systems and their computational expressiveness. For example, 2-dimensional PCD systems cannot simulate non-deterministic automata having a non-planar transition graph.

There have been other works on simulation of transition systems by dynamical systems. For example, in [9] the boolean transition function of an automaton, defined over $\{0,1\}^m$, is realized by its continuous extension to

$[0,1]^m$ using arithmetical operations. Similarly stacks have been simulated by rational arithmetic in [8]. In these works, however, the simulating system is *already defined over discrete time*, i.e., using iterated maps of the form $\mathbf{x}_{n+1} = f(\mathbf{x}_n)$. Our construction, on the other hand, uses continuous-time systems.

There have been various undecidability results for other variants of hybrid systems with piecewise-constant derivatives (*timed automata* [1] or *integration graphs* [12]), but those were obtained in a richer model where a transition between regions is accompanied by a discrete change, and the trajectories are discontinuous. In [6] automata were simulated (without a precise formal definition of this term) by smooth dynamical systems defined over a state-space of certain symmetric matrices. Those systems have high dimensionality that grows with the size of the automaton. Recently Branicky [5] used differential equations with continuous vector fields, as well as several models of hybrid systems, to simulate Turing machines. The dynamical systems considered in [5] have, informally speaking, infinitely many regions (or components) unlike our PCD systems.

The closest work to ours has been reported in [17] where stack machines were constructed from optical elements such as mirrors and lenses. These constructions were used to prove undecidability of the ray tracing problem. It should be noted, however, that optical systems, as well as billiard models, require a richer model, where the phase-space is $2n$-dimensional (the velocity in each spatial dimensions is also a state variable) and the trajectories are discontinuous in this phase-space (the velocity goes through an abrupt change). Hence the equivalence between our PCD results and theirs is an optical illusion.

Finally the philosopher Putnam [15], while attempting to "prove" the thesis *every open physical system realizes every automaton*, uses a notion of aimulation we find implausible. Consider, for example a deterministic automaton without input, generating the sequence $(q_1 q_2)^\omega$. Then the dynamical system $\frac{dx}{dt} = 1$ (or any other system with a non-cyclic behavior) simulates the automaton by letting $\pi(x) = q_1$ when $2i < x < 2i+1$, and $\pi(x) = q_2$ when $2i+1 < x < 2i+2$ for any integer $i \geq 0$. This simulation works only if we consider a fixed initial state (otherwise we need a different abstraction for each state) and, moreover, π is topologically rather complex: $\pi^{-1}(q)$ is a union of infinitely many disconnected sets, which contradicts our intuition concerning abstractions.

Acknowledgements

We would like to thank A. Bouajjani, S. Yovine and J. Sifakis for many helpful comments. In particular A. Bouajjani contributed to an earlier undecidability result for 6 dimensions that inspired the current construction.

References

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

[2] A. Asarin and O. Maler, On some Relations between Dynamical Systems and Transition Systems, in S. Abiteboul and E. Shamir, editors, *Proc. of ICALP'94*, Lect. Notes in Comp. Sci. 820, pages 59–72, Springer-Verlag, 1994.

[3] R. Alur, C. Courcoubetis, and D.L. Dill, Model checking in dense real time, Information and Computation 104, 2–34, 1993.

[4] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, Hybrid automata: An algorithmic approach to the specification and analysis of hybrid systems. In R.L Grossman, A. Nerode, A. Ravn and H. Rischel, editors, *Hybrid Systems*, Lect. Notes in Comp. Sci. 736, pages 209–229, Springer-Verlag, 1993.

[5] M.S. Branicky, Universal Computation and other Capabilities of Hybrid and Continuous Dynamical Systems, This issue.

[6] R.W. Brockett, Smooth dynamical systems which realize arithmetical and logical operations, In H. Nijmeijer and J.M. Schumacher, editors, *Three Decades of Mathematical Systems Theory*, Lect. Notes in Control and Information Sciences, pages 19–30, Springer-Verlag, 1989.

[7] A. Brondsted, *An Introduction to Convex Polytopes*, Springer-Verlag, New-York, 1983.

[8] M. Cosnard, M. Garzon and P. Koiran, Computability properties of low-dimensional dynamical systems, In P. Enjalbert, A. Finkel

and K.W. Wagner, editors, *Proc. of the 10th Ann. Symp. on Theoretical Aspects of Computer Science*, Lect. Notes in Comp. Sci. 665, pages 365–373, Springer-Verlag, 1993.

[9] R.A. DeMillo and R.J. Lipton, Defining software by continuous smooth functions, *IEEE Trans. on Software Engineering*, Vol. 17, No. 4, 1991.

[10] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, Symbolic model-checking for real-time systems, *Information and Computation* 111, 193–244, 1994.

[11] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979.

[12] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine, Integration graphs: A class of decidable hybrid systems, In R.L Grossman, A. Nerode, A. Ravn and H. Rischel, editors, *Hybrid Systems*, Lect. Notes in Comp. Sci. 736, pages 179–208, Springer-Verlag, 1993.

[13] O. Maler, Z. Manna, and A. Pnueli, From timed to hybrid systems, In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, Lect. Notes in Comp. Sci. 600, pages 447–484. Springer-Verlag, 1992.

[14] O. Maler and A. Pnueli, Reachability analysis of planar multi-linear systems, In C. Courcoubetis, editor, *Proc. of the 5th Workshop on Computer-Aided Verification*, Lect. Notes in Comp. Sci. 697, pages 194–209. Springer-Verlag, 1993.

[15] H. Putnam. *Representation and Reality*, MIT Press, Cambridge, MA, 1988.

[16] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, An approach to the description and analysis of hybrid systems, In R.L Grossman, A. Nerode, A. Ravn and H. Rischel, editors, *Hybrid Systems*, Lect. Notes in Comp. Sci. 736, pages 149–178, Springer-Verlag, 1993.

[17] J.H. Reif, J.D. Tygar, and A. Yoshida, The computability and complexity of optical beam tracing, in *Proc. 31st Annual Symposium on Foundations of Computer Science*, St. Louis, Missouri, 106–114, IEEE Press 1990.