Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy*

Eugene Asarin
Institute for Information Transmission Problems
19 Bol. Karetnyi per.
101447 Moscow, Russia
asarin@ippi.ac.msk.su

Oded Maler
VERIMAG-CNRS
Centre Equation
2, av. de Vignate
38610 Gières, France
Oded.Maler@imag.fr

May 10, 1997

*A preliminary version of the paper appeared in P.S. Thiagarajan (Ed.), *Proc. FST/TCS'95*, 471-483, LNCS 1026, Springer 1995. This research was supported in part by the European Community projects HYBRID EC-US-043 and INTAS-94-697 as well as by Research Grant #93-012-884 of Russian Foundation for Basic Research. VERIMAG is a joint laboratory of CNRS, INPG and UJF.

Running head: *Achilles and the Tortoise*

Addresss of contact author:

Oded Maler
VERIMAG
Centre Equation
2, av. de Vignate
38610 Gières, France

E-mail: Oded.Maler@imag.fr
Phone: +33 4 76 63 48 41
Fax: +33 4 76 63 48 50

Abstract

In this paper we show how to construct for every set P of integers in the arithmetical hierarchy a dynamical system $\mathcal{H}$ with piecewise-constant derivatives (PCD) such that deciding membership in P can be reduced to solving the reachability problem between two rational points for $\mathcal{H}$. The ability of such apparently-simple dynamical systems, whose definition involves only *rational* parameters, to "solve" highly unsolvable problems is closely related to Zeno's paradox, namely the ability to pack infinitely many discrete steps in a bounded interval of time.

1 Introduction

This research was initially motivated by recent attempts to apply program verification methodology to *hybrid dynamical systems*, i.e. systems combining discrete and continuous components (see, for example, [ACH+95], [AKNS95]). In [AMP95] we have introduced *PCD systems*, a sub-class of the so-called linear hybrid automata of [ACH+95]. In such systems, Euclidean space is partitioned into convex polyhedra ("regions") and a constant "slope" is assigned to every region. We have shown in [AMP95] that for two-dimensional PCD systems the reachability problem (whether there exists a trajectory between two rational points) is decidable while the same problem becomes undecidable when we move to three dimensions or more. The undecidability is due to the ability of 3-dimensional PCDs to simulate any Turing machine.

From the verification point of view, this paper worsens our previous negative result: it shows that by adding each time few dimensions we can climb up indefinitely in the arithmetical hierarchy. This means that such hybrid systems are, at least in the worst-case sense, "very very hard" to analyze. From the dual perspective of expressiveness, this paper offers an alternative geometrical model of computation, that is very simple, yet powerful enough to solve any arithmetical problem. The constructions used in this paper show some interesting connections between dynamics and computation, and may contribute to a better understanding of both.

The rest of the paper is organized as follows: in section 2 we introduce the arithmetical hierarchy, PCD systems and define recognition (acceptance) by such systems. In section 3 we give a short review of our previous results concerning the recognition of r.e. sets by 3-dimensional PCDs and draw the plan for the proof of the main result. In section 4 we prove a key lemma allowing to construct from a PCD $\mathcal{H}$ that semi-recognizes a set P a PCD $\mathcal{H}'$ that fully recognizes P. Two additional lemmas and the final result are presented in section 5, followed by a discussion in section 6.

2 Preliminaries

Let $X = \mathsf{R}^d$ for some d. We use boldface letters to denote points (vectors) taken from X. A convex *polyhedral set* is a subset of X consisting of all $\mathbf{x}$

satisfying a finite number of inequalities of the form $\mathbf{a} \cdot \mathbf{x} \leq b$ or $\mathbf{a} \cdot \mathbf{x} < b$.

PCD Systems

Definition 1 (PCD System) *A piecewise-constant derivative (PCD) system is a dynamical system $\mathcal{H} = (X, f)$ where X is the state-space and 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 $\boldsymbol{c} \in C$, $f^{-1}(\boldsymbol{c})$ is a finite union of convex polyhedral sets.*

A *trajectory* of $\mathcal{H}$ starting at some $\mathbf{x}_0 \in X$ is a solution of the differential equation

$$\frac{d^+\mathbf{x}}{dt} = f(\mathbf{x}) \tag{1}$$

with initial condition $\mathbf{x} = \mathbf{x}_0$, that is, a continuous function $\xi : \mathsf{R}^+ \to X$ such that $\xi(0) = \mathbf{x}_0$ and for every t, $f(\xi(t))$ is defined and is equal to the right derivative of $\xi(t)$.

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 (see figure 1). The trajectories of such systems are broken lines, with the breakpoints occurring on the boundaries of the regions. A PCD is *bounded* if the domain of f is a bounded subset of X.

It is important to emphasize that we assume that all constants in the system's definition (inequalities, derivatives, initial points) are *rational*. Hence the expressive power of PCD is *not* achieved using the introduction of some uncomputable real numbers.

The Arithmetical Hierarchy

We review here some classical definitions from recursion theory (see [R67]). The arithmetical hierarchy consists of the classes $\Sigma_1, \Sigma_2, \ldots$ and $\Pi_1, \Pi_2, \ldots$ of sets of integers defined inductively as follows: Σ_1 consists of all the sets $P \subseteq \mathsf{N}$ such that there exists a Turing machine that halts on an input n if and only if $n \in P$. The class Π_i consists of all the sets P such that $\overline{P} \in \Sigma_i$

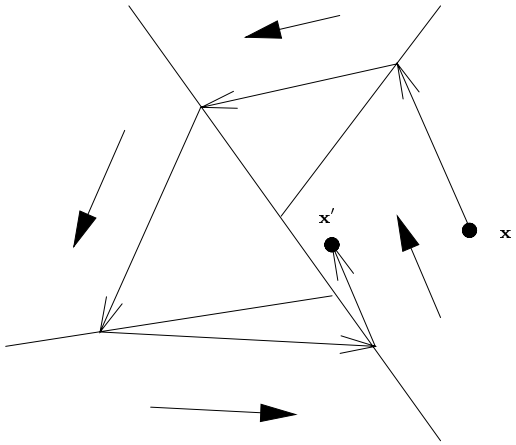

Figure 1: A 2-dimensional PCD system with 4 regions and an initial segment of a trajectory from $\mathbf{x}$ to $\mathbf{x}'$.

and Σ_{i+1} is the class of all sets P defined as $P = \{n : \exists m \langle m, n \rangle \in P'\}$ for some $P' \in \Pi_i$, where $\langle\rangle$ is some computable pairing function.[1]

The arithmetical hierarchy is infinite and it satisfies the strict inclusions $\Pi_i \subset \Sigma_{i+1}$ and $\Sigma_i \subset \Pi_{i+1}$. The class Σ_1 is sometimes called the class of recursively enumerable (r.e.) sets. Due to the undecidablity of the halting problem for Turing machines, membership of some $n \in \mathsf{N}$ in some $P \in \Sigma_1$ is only semi-decidable, i.e., there is an algorithm which is guaranteed to terminate if $n \in P$ but no algorithm will terminate for every n and P when $n \notin P$.

Recognition by PCDs

We will use PCD systems as recognizers of sets of integers as follows:

Definition 2 (Recognition by PCDs) *A PCD recognizer is a tuple* $\widehat{\mathcal{H}} = (R^d, f, I, r, \boldsymbol{x}^{\mathrm{A}}, \boldsymbol{x}^{\mathrm{R}})$ *where:*

- $\mathcal{H} = (R^d, f)$ *is a PCD,*
- $I = [0,1] \times \{0\}^{d-1}$ *is a one-dimensional subset of* X *(the "input port"),*

[1] A pairing function is a bijection from $\mathsf{N} \times \mathsf{N}$ to N, a standard way to encode two dimensions using one.

- $r : \mathsf{N} \to [0,1] \cap \mathsf{Q}$ *is a recursive injective coding function, and*
- $\boldsymbol{x}^{\mathrm{A}}, \boldsymbol{x}^{\mathrm{R}} \in R^d - I$ *are two distinct points (accepting and rejecting states).*

We assume that $f(\boldsymbol{x}^{\mathrm{A}}) = f(\boldsymbol{x}^{\mathrm{R}}) = 0$.

The system $\widehat{\mathcal{H}}$ semi-recognizes a set $P \subseteq \mathsf{N}$ iff for every n, the trajectory starting at $(r(n), 0, \ldots, 0)$ can continue forever[2] and that it eventually reaches $\mathbf{x}^{\mathrm{A}}$ iff $n \in P$. We say that $\hat{\mathcal{H}}$ (fully) recognizes P when, in addition, this trajectory reaches $\mathbf{x}^{\mathrm{R}}$ iff $n \notin P$.

In other words, every integer n is encoded into a distinct rational point on the input port, and the membership of n in P is indicated by whether the trajectory starting at this point settles in the accepting (rejecting) point after a *finite* amount of time.

Remark: This notion of recognition is not much different from recognition (acceptance) of sets by Turing machines. A TM is nothing but a discrete dynamical system whose state-space is the set of all its configurations (state, tape, head position). This system accepts an input n if the trajectory starting at a configuration where n is encoded on the tape eventually reaches the halting state.

3 PCDs Realize TMs

In this section we review some of the definitions and results of [AMP95] concerning the realization of Turing machines by PCD systems, or more generally the realization of discrete transition systems by continuous dynamical systems. By a transition systems we mean $\mathcal{A} = (Q, \delta)$ where Q is a countable set of states and $\delta : Q \to Q$ is a transition function.

Definition 3 (Realization of Transition Systems) *A PCD* $\mathcal{H} = (X, f)$ *realizes a transition system* $\mathcal{A} = (Q, \delta)$ *if there exists an injective and surjective partial function* $\pi : X \to Q$ *such that* $\delta(q) = q'$ *iff there is a trajectory of* $\mathcal{H}$ *from* $\pi^{-1}(q)$ *to* $\pi^{-1}(q')$ *that does not intersect the domain of* π *between these two points.*

[2] Which means that it always stays inside the domain of definition of f, and that it can continue from every point.

The realization result uses the equivalence between TMs and two-stack machines. A stack is an element of Γ^* where $\Gamma = \{0, 1\}$. We define the following two functions: PUSH: $\Gamma \times \Gamma^* \to \Gamma^*$ and POP: $\Gamma^* \to \Gamma \times \Gamma^*$ as $\text{PUSH}(v, S) = vS$ and $\text{POP}(vS) = (v, S)$, where vS denotes the concatenation of the symbol v and the string S.

Definition 4 (2PDA) *A deterministic two-stack pushdown-automaton (2PDA) is a transition system $\mathcal{A} = (Q \times \Gamma^* \times \Gamma^*, \delta)$ for some $Q = \{q_1, \ldots q_n\}$ such that δ is defined using a finite collection of statements of one of the following two forms:*

$$\begin{array}{lll} q_i\text{:} & S_\alpha := \text{PUSH}(v, S_\alpha); & \\ & \text{GOTO } q_j & \end{array} \qquad \begin{array}{lll} q_i\text{:} & (v, S_\alpha) := \text{POP}(S_\alpha); & \\ & \text{IF } v = 0 \text{ GOTO } q_{i_0}; & \\ & \text{IF } v = 1 \text{ GOTO } q_{i_1}; & \end{array}$$

where $\alpha \in \{1, 2\}$.

The contents of a stack is denoted by $S = s_1 s_2 \ldots$ where s_1 is the top of the stack. We define an encoding function $\rho : \Gamma^* \to [0, 1]$ as

$$\rho(S) = \sum_{i=1}^{|S|} s_i 2^{-i}.$$

It is easily verified that the stack operations have arithmetic counterparts that operate on the representation:

$$\begin{array}{rcll} S' = & \text{PUSH}(v, S) & \text{iff} & \rho(S') = (\rho(S) + v)/2 \\ (S', v) = & \text{POP}(S) & \text{iff} & \rho(S') = 2\rho(S) - v \end{array}$$

Remark: We present here a simplified version of the construction in [AMP95], omitting some tedious details concerning the encoding of rational numbers using bottom-less stacks.

Theorem 1 (Realization of 2PDAs [AMP95]) *Every 2PDA can be realized by a 3-dimensional bounded PCD system.*

Sketch of Proof: We show first how one-stack PDAs are realized. Consider the three two-dimensional sub-systems depicted in figure 2 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:

$x' = (x+1)/2$ PUSH 1
$x' = x/2$ PUSH 0
$x' = 2x - 1/2$ POP

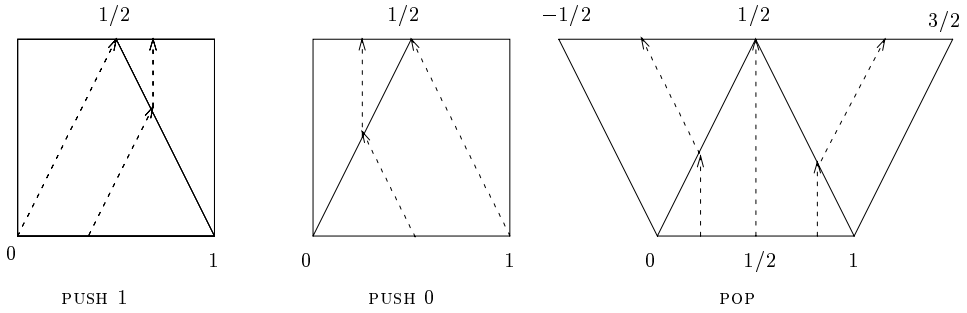

Figure 2: The basic elements.

If $x = \rho(S)$ at the "input port" ($y = 0$) of a PUSH element, then $x' = \rho(S')$ 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 $x' = \rho(S') - 1/2$, otherwise it goes to the right port with $x' = \rho(S') + 1/2$. 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 3). Finally the state-mapping is defined as $\pi(x, y, z) = (q_i, S)$ iff $y = z = 0$, $2i \leq x < 2i + 1$ and $\rho(S) = x - 2i$.

This construction generalizes naturally to 2PDAs. We define an encoding function $\bar{\rho} : \Gamma^* \times \Gamma^* \to [0,1] \times [0,1]$ by letting $\bar{\rho}(S_1, S_2) = (\rho(S_1), \rho(S_2))$. 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 4. From this we can immediately conclude that

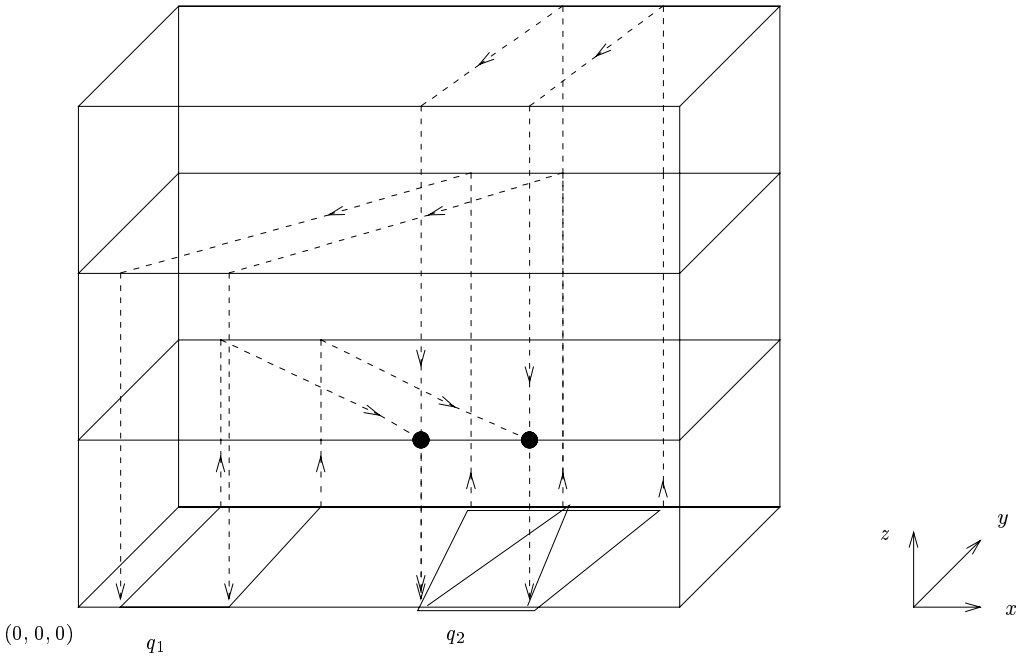

Figure 3: Simulating a PDA with 2 states, defined by: $q_1 : S :=\text{PUSH}(1, S)$; GOTO q_2; $q_2 : (v, S) :=\text{POP}(S)$; If $v = 1$ THEN GOTO q_2 ELSE GOTO q_1.

a 2PDA can be realized by a 4-dimensional PCD.[3] ▟

Corollary 2 (PCD and Σ_1) *Every Σ_1 set P is semi-recognized by some 3-dimensional bounded PCD.*

Proof: We take the 2PDA associated with P and assume w.l.o.g. that whenever it halts, it halts at a given configuration (q, S_1, S_2). By constructing $\mathcal{H}$ as in the proof of theorem 1, encoding N into the input port and letting $\mathbf{x}^{\mathrm{A}} = \pi(q, S_1, S_2)$ we obtain a semi-recognizing PCD. ▟

Corollary 3 (PCD and Recursive Functions) *Every recursive function $\varphi : \mathsf{N} \times \mathsf{N} \to \mathsf{N}$ can be computed*[4] *by a 3-dimensional bounded PCD.*

[3] In [AMP95] we introduced additional tricks to avoid the fourth dimension, but this paper deals with the infinite and small constants do not matter.

[4] Computing a function by a PCD is a natural extension of deciding membership in a set. You just introduce an output port and use r^{-1} to decode the result.

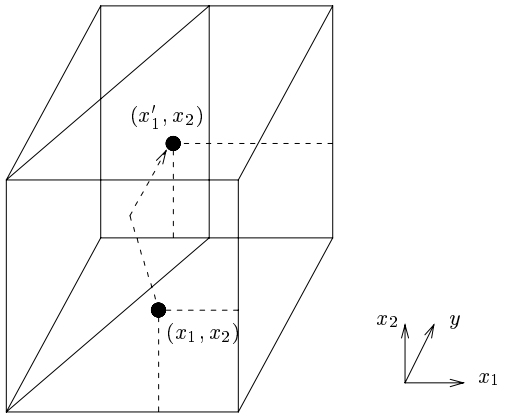

Figure 4: An element simulating the operation $\textsc{push}(0, S_1)$

Proof: We take the 2PDA that computes φ where the inputs are written each on one stack and the output is written on S_1 and use theorem 1. ∎

Corollary 2 gives us the basis for climbing up the hierarchy. In order to continue we need the following lemmata ordered according to their decreasing difficulty:

1. From a PCD that semi-recognizes P one can construct a PCD that recognizes P.
2. From a PCD that recognizes P one can construct
   (a) a PCD that semi-recognizes $\{x : \exists y \, \langle x, y \rangle \in P\}$, and
   (b) a PCD that recognizes $\overline{P}$.

4 From Semi-recognition to Recognition

The intuitive idea behind the first lemma is the following. Suppose $\mathcal{H}$ semi-recognizes P. The trajectory corresponding to some $n \notin P$ is wandering forever in R^d without reaching $\mathbf{x}^{\mathrm{A}}$. We will create a higher dimensional PCD $\mathcal{H}'$ such that $\mathcal{H}'$ "mimics" $\mathcal{H}$ (in the projection) for a unit interval, then it goes to some other regions, and comes back after having divided all the variables by 2. Then it mimics $\mathcal{H}$ again on a *smaller scale* for a temporal interval of length 1/2, then divides the variables by 2 and so on. Clearly

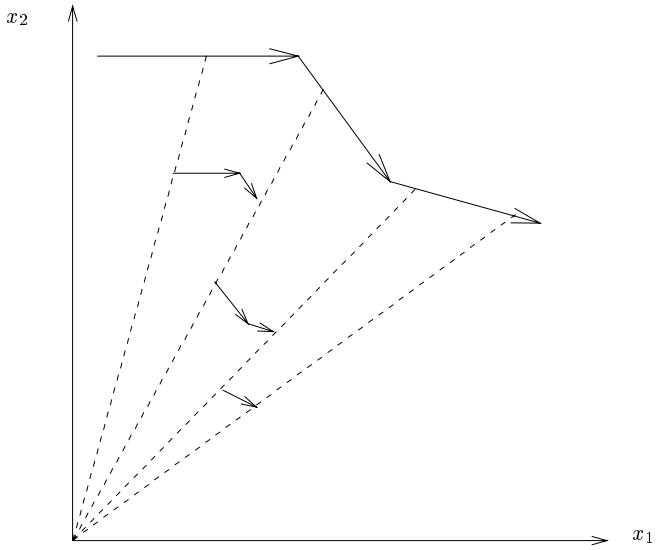

Figure 5: A 2-dimensional trajectory in $\mathcal{H}$ and the projection of its "Zenonified" corresponding trajectory in $\mathcal{H}'$.

every diverging trajectory in $\mathcal{H}$ will reach the origin $(0,\ldots,0)$ in $\mathcal{H}'$ after a *finite* amount of time[5] (and an infinite number of region switchings) and thus $\mathcal{H}'$ fully recognizes P (see figure 5).

In order to gain some intuition for "PCD programming" let us first build a PCD that divides a single variable x by 2 using an auxiliary variable y initialized to 0. The system is depicted graphically in figure 6 and its syntactic description is given by the following set of "guarded commands":

$$\begin{array}{lrcl} A: & x>0 \wedge y \geq 0 & \longrightarrow & \dot{x}=-1,\ \dot{y}=1/2 \\ B: & x\leq 0 \wedge y > 0 & \longrightarrow & \dot{x}=-1,\ \dot{y}=-1 \\ C: & x<0 \wedge y \leq 0 & \longrightarrow & \dot{x}=1,\ \dot{y}=-1 \\ D: & x\geq 0 \wedge y < 0 & \longrightarrow & \dot{x}=1,\ \dot{y}=1 \end{array}$$

It can be easily verified that whenever started at some point $(x,0)$, the system completes one cycle and returns to $(x/2,0)$. The time to complete such a cycle is $2.5x$ and if we make k cycles we arrive at $(2^{-k}x,0)$ within $2.5x\sum_{i=0}^{k-1}2^{-i}$ time. If we let $f(0,0)=(0,0)$, the trajectory starting at $(x,0)$,

[5] This resembles a Turing machine which doubles its speed every step.

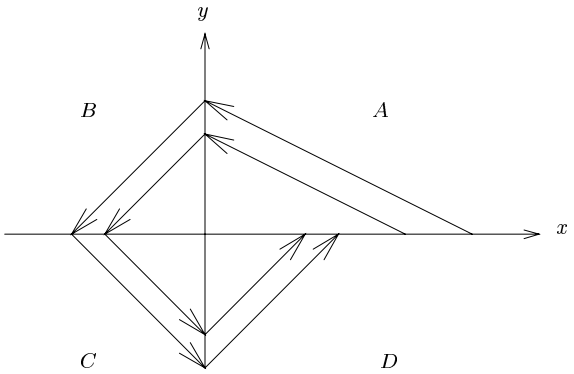

Figure 6: A 2-dimensional PCD for dividing a 1-dimensional quantity.

spiralling infinitely many times in A, B, C, D during the temporal interval $[0, 5x)$ and staying in $(0, 0)$ in the interval $[5x, \infty)$, is indeed a valid trajectory of the system according to definition 1, i.e. a solution to the initial value problem of the differential equation (1). As we will see later, this can be generalized to d variables using $4d$ regions and $d + 1$ dimensions.

The functionality of the division system can be captured by the following informal sequential pseudo-code:

repeat
A: $y := x/2$; $x := 0$
B: $x := -y$; $y := 0$
C: $y := x$; $x := 0$
D: $x := -y$; $y := 0$
until $x = 0 \wedge y = 0$

The last construction we need before we prove the main lemma is the *homogenization* of a PCD (and a dynamical system in general). A dynamical system is homogenous if it has the following property: if there is a trajectory segment from $\mathbf{x}$ to $\mathbf{x}'$ which takes t time, then, for every $k \in (0, 1]$ there exists a similar (in the geometrical sense) trajectory from $k\mathbf{x}$ to $k\mathbf{x}'$ which takes kt time.

Any PCD $\mathcal{H} = (\mathsf{R}^d, f)$ can be converted into a homogenous system $\mathcal{H}^0 = (\mathsf{R}^{d+1}, f^0)$ such that $\mathcal{H}$-reachability from $\mathbf{x}$ to $\mathbf{x}'$ implies $\mathcal{H}^0$-reachability from

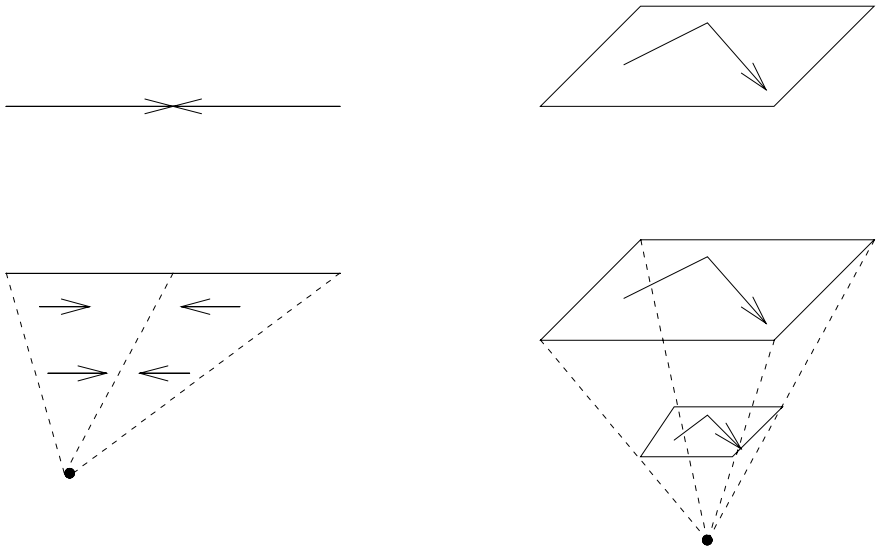

Figure 7: A homogenization of a 1-dimensional system (left) and a 2-dimensional system (right).

$(x_{d+1}\mathbf{x}, x_{d+1})$ to[6] $(x_{d+1}\mathbf{x}', x_{d+1})$ for every $x_{d+1} \in [0,1]$.

Geometrically $\mathcal{H}^0$ is obtained from $\mathcal{H}$ by choosing a point as the origin and replacing every d-dimensional region by a $d+1$-dimensional "pyramid" rooted at that point (see figure 7). Syntactically all you do is replace every inequality of the form $\mathbf{a} \cdot \mathbf{x} \leq b$ in $\mathcal{H}$ by

$$(\mathbf{a}, 0) \cdot (\mathbf{x}, x_{d+1}) \leq b x_{d+1}$$

and add the conjunct $0 \leq x_{d+1} \leq 1$ to every definition of a region. Finally every slope $\mathbf{c}$ is replaced by $(\mathbf{c}, 0)$.

Lemma 4 (Semi-Recognition $\Rightarrow$ Recognition) *From a bounded PCD $\mathcal{H} = (R^d, f, I, r, \boldsymbol{x}^{\mathrm{A}}, \boldsymbol{x}^{\mathrm{R}})$ semi-recognizing P one can construct a bounded PCD $\mathcal{H}' = (R^{d+3}, f', I', r', \boldsymbol{x}'^{\mathrm{A}}, \boldsymbol{x}'^{\mathrm{R}})$ that recognizes P.*

Proof: Without loss of generality we can assume that our PCDs always work in a bounded subset of R^d_+ sufficiently far from the origin.[7] We augment the

[6] We slightly abuse notations: for $\mathbf{x} = (x_1, \ldots, x_d)$ we use $(\mathbf{x}, x_{d+1})$ to denote $(x_1, \ldots, x_d, x_{d+1})$.

[7] because of boundedness the whole system can be translated to the positive side.

system with three additional variables: x_{d+1}, h and y. The first of these variables, x_{d+1}, serves for homogenization and is treated as any other x_i during the division phase. The overall behavior of the system can be captured using the following pseudo-code:

repeat
  SimulateH;
  if (result=Accept)
    then return(Accept);
  DivideXby2;
until $\mathbf{x} = 0$;
return(Reject)

SimulateH stands for the simulation of the behavior of a homogenized version of $\mathcal{H}$ for a duration determined by x_{d+1}, using h as a timer to measure that time. If $\mathcal{H}$ reaches its accepting state, the simulation terminates, returning a positive result. Otherwise, DivideXby2 divides all the variables by 2 and simulates $\mathcal{H}$ for $x_{d+1}/2$ time and so on. The division procedure, described using the pseudo-code below, uses the auxiliary variable y:

$\{$*initially* $\mathbf{x} = \mathbf{x}_0, y = 0$ $\}$
for $k := 1$ **to** $d + 1$
  do
    A_k: $y := (-1)^{k+1} x_k/2; x_k := 0;$
    B_k: $x_k := (-1)^k y; y := 0;$
  od;
$\{$*now* $\mathbf{x} = -\mathbf{x}_0/2, y = 0$ $\}$
Z: $h := 0$; $\{$*reset the "timer"* h $\}$
for $k := 1$ **to** $d + 1$
  do
    C_k: $y := (-1)^{d+k+1} x_k/2; x_k := 0;$
    D_k: $x_k := (-1)^{d+k} y; y := 0;$
  od;
$\{$*finally* $\mathbf{x} = \mathbf{x}_0/2, y = 0, h = 0$ $\}$

The regions of $\mathcal{H}'$ are constructed as follows. Every original region of $\mathcal{H}$ is homogenized and the conditions $y = 0$ and $h < x_{d+1}$ are added. All these

regions also satisfy $x_1, \ldots, x_{d+1} > 0$. In addition to the original derivatives we have $(\dot{x}_{d+1}, \dot{h}, \dot{y}) = (0, 1, 0)$. Therefore, whenever the system enters such a region with $x_{d+1} = c$ it simulates the original system for c time (see the first phase, denoted by $\mathcal{H}$, in the signal diagram of figure 8). The detailed definition of the new regions is given below for every k, $1 \leq k \leq d+1$ (only non-zero derivatives are written down):

$$A_k : \begin{cases} x_1, \ldots, x_{k-1} & < & 0 \\ x_{k+1}, \ldots, x_{d+1} & > & 0 \\ x_k & > & 0 \\ (-1)^k y & \leq & 0 \end{cases} \qquad \longrightarrow \qquad \dot{x}_k = -1,\ \dot{y} = (-1)^{k+1}/2$$

Additional condition for A_1 : $\quad h = x_{d+1}$

$$B_k : \begin{cases} x_1, \ldots, x_{k-1} & < & 0 \\ x_{k+1}, \ldots, x_{d+1} & > & 0 \\ x_k & \leq & 0 \\ (-1)^k y & < & 0 \end{cases} \qquad \longrightarrow \qquad \dot{x}_k = -1,\ \dot{y} = (-1)^k$$

$$C_k : \begin{cases} x_1, \ldots, x_{k-1} & > & 0 \\ x_{k+1}, \ldots, x_{d+1} & < & 0 \\ x_k & < & 0 \\ (-1)^{d+k+1} y & \leq & 0 \end{cases} \qquad \longrightarrow \qquad \dot{x}_k = 1,\ \dot{y} = (-1)^{d+k+1}$$

Additional condition for C_1 : $\quad h = 0$

$$D_k : \begin{cases} x_1, \ldots, x_{k-1} & > & 0 \\ x_{k+1}, \ldots, x_{d+1} & < & 0 \\ x_k & \geq & 0 \\ (-1)^{d+k+1} y & > & 0 \end{cases} \qquad \longrightarrow \qquad \dot{x}_k = 1,\ \dot{y} = (-1)^{d+k}$$

In addition we add a special region Z (for resetting h to 0):

$$\mathrm{Z} : \begin{cases} x_1, \ldots, x_{d+1} & < & 0 \\ y & = & 0 \\ h & > & 0 \end{cases} \qquad \longrightarrow \qquad \dot{h} = -1$$

For every k, a passage through the sequence of regions A_k, B_k, C_k, D_k would result in dividing x_k by 2. However we do first $A_1, B_1, A_2, B_2, \ldots, A_{d+1}, B_{d+1}$ making all variables negative, then we enter Z, reset h to zero and then complete $C_1, D_1, C_2, D_2, \ldots, C_{d+1}, D_{d+1}$.

The reader should verify the example in figure 8 for $d = 1$, where (x_1, x_2) start the division phase at $(1, 1)$ and terminate it at $(1/2, 1/2)$, ready to simulate the (scaled-down) behavior of $\mathcal{H}$, now for $1/2$ time interval. Clearly the trajectory of $\mathcal{H}'$ converges in finite time to $(0, \ldots, 0)$ which we can consider as the rejecting point $\mathbf{x}'^{\mathrm{R}}$. We should take care of not treating accepting trajectories (those that reach $\mathbf{x}^{\mathrm{A}}$ in $\mathcal{H}$) this way. This is done by adding the condition $\mathbf{x} \neq \mathbf{x}^{\mathrm{A}} x_{d+1}$ to each of the $\mathcal{H}$-regions defined above. Then when $\mathbf{x} = \mathbf{x}^{\mathrm{A}} x_{d+1}$ we have two additional regions: if $h > 0$ we just lower h to zero. When $h = 0$ we let $\dot{\mathbf{x}} = \mathbf{x}^{\mathrm{A}}$ and $\dot{x}_{d+1}$=1 until $x_{d+1} = 1$. This way we reach the point $(\mathbf{x}^{\mathrm{A}}, 1, 0, 0)$ which is the new accepting point $\mathbf{x}'^{\mathrm{A}}$. ▟

5 Quantifier Elimination

Lemma 5 (Quantifier Elimination) *Let $\mathcal{H}$ be a bounded PCD in $\mathbb{R}^d$ that recognizes a set P. Then one can construct a bounded PCD $\widetilde{\mathcal{H}}$ in $\mathbb{R}^{d+2}$ that semi-recognizes the set $\widetilde{P} = \{n : \exists m \langle n, m \rangle \in P\}$.*

Proof: The idea of the proof is standard: given n we just test one after the other all the possible values of m and use the PCD $\mathcal{H}$ with inputs $\langle n, m \rangle$ to verify whether these inputs belong to P. For any input $n \in \widetilde{P}$ we will eventually find a good m while for $n \notin \widetilde{P}$ the process will continue forever. This is captured by the pseudo-code:

input(n);
$m := 0$;
repeat
  if $\mathcal{H}(\langle n, m \rangle)$ =Accept
    then
      return(Accept)
    else
      $m := m + 1$
forever

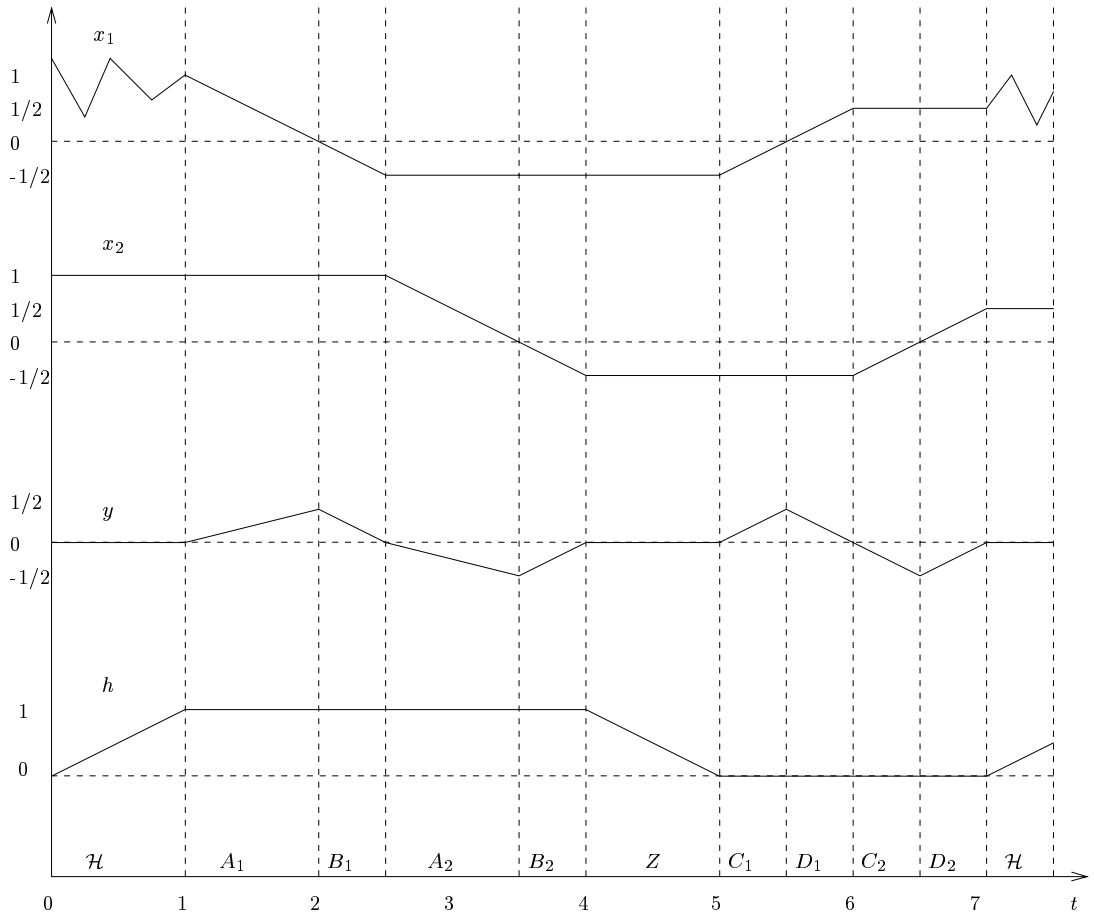

Figure 8: An initial part of the behavior of $\mathcal{H}'$ when $d = 1$. The regions through which the system passes are written below ($\mathcal{H}$ denotes any region in which $\mathcal{H}'$ simulates $\mathcal{H}$).

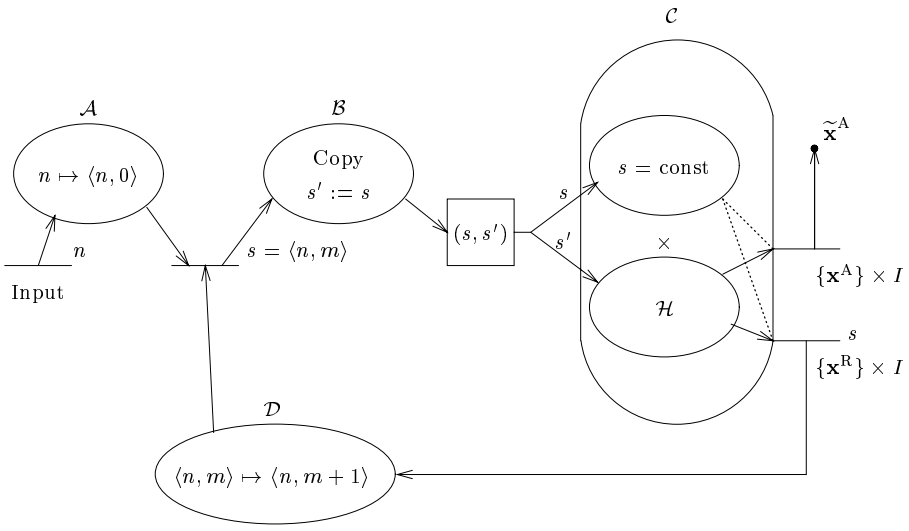

Figure 9: The PCD $\widetilde{\mathcal{H}}$ semi-recognizing $\widetilde{P} = \{n : \exists m \langle n, m\rangle \in P\}$

In order to avoid an unreadable collection of linear inequalities we will describe the PCD $\widetilde{\mathcal{H}}$ more schematically – see figure 9. Bold line segments and squares in the figure stand for one- and two-dimensional ports respectively; arrows denote connections. Ellipses stand for PCDs that compute various recursive integer functions (based on corollary 3).

The only block that needs a special description is $\mathcal{C}$. This block is a direct product of $\mathcal{H}$ and the line segment $I = [0, 1]$. When an input (s, s') is provided to this block the variable s is preserved unchanged (for future use) and s' is used as an input for a copy of $\mathcal{H}$. So the trajectory exits $\mathcal{C}$ either at $\{\mathbf{x}^{\mathrm{A}}\} \times I$ (if $\mathcal{H}$ accepts s') or at $\{\mathbf{x}^{\mathrm{R}}\} \times I$ otherwise.

Let us describe the trajectory entering $\widetilde{\mathcal{H}}$ through the input port at a point $r(n)$. First $s = \langle n, 0\rangle$ is calculated by $\mathcal{A}$. The block $\mathcal{B}$ creates another copy of s denoted by s' (that is, the trajectory exits $\mathcal{B}$ through its two-dimensional output port at the point $(r(s), r(s))$). This copy is used as input for the original PCD $\mathcal{H}$ in block $\mathcal{C}$. Meanwhile s is preserved for further use. If $\langle n, 0\rangle \in P$, the trajectory exits $\mathcal{C}$ at $\{\mathbf{x}^{\mathrm{A}}\} \times I$ and then goes to the new accepting point $\widetilde{\mathbf{x}}^{\mathrm{A}}$. In the case when $\langle n, 0\rangle \notin P$ further search is necessary and the trajectory goes from $\{\mathbf{x}^{\mathrm{R}}\} \times I$ to the block $\mathcal{D}$ which transforms $\langle n, 0\rangle$ to $\langle n, 1\rangle$. This last value is used at the next iteration of the loop ad infinitum.

Recall that all these blocks are recursive and can be realized according to corollary 3.

If $n \in \widetilde{P}$ then an m satisfying $\langle n, m \rangle \in P$ exists and the m^{th} iteration of the main loop the PCD $\widetilde{\mathcal{H}}$ will stop in the accepting point $\tilde{\mathbf{x}}^{\mathrm{A}}$. Otherwise $\widetilde{\mathcal{H}}$ will check all the natural m's in turn and will never halt. Hence $\widetilde{\mathcal{H}}$ does semi-recognize the set $\widetilde{P}$. The system $\widetilde{\mathcal{H}}$ fits in $d+2$ dimensions. In fact its largest block is $\mathcal{C}$ which uses d dimension for $\mathcal{H}$, one dimension for s and one more dimension for merging incoming and outgoing connections. ▟

Lemma 6 (Complementation) *From a PCD that recognizes P one can construct a PCD that recognized $\mathsf{N} - P$.*

Proof: Exchange $\mathbf{x}^{\mathrm{A}}$ and $\mathbf{x}^{\mathrm{R}}$. ▟
From this we conclude:

Theorem 7 (Main Result) *Every set P in the arithmetical hierarchy can be recognized by a PCD system of finite dimension and a finite number of regions.*

The number of dimensions used to recognize $P \in \Sigma_i \cup \Pi_i$ is $5i + 1$.

6 Discussion

What is the significance of this result? On the one hand we have a rather simple class of dynamical systems which are "locally 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. This is unlike more general continuous dynamical systems where one can only *approximate* trajectories numerically. On the other hand these systems give rise to highly undecidable reachability problems. The reasons for this "expressiveness excess" of the model should worry researchers in hybrid systems and urge them to find ways to tackle these problems, either by restricting the models or by changing the questions. One common solution is to exclude Zeno trajectories from the semantics of the system which brings back the reachability problem into the level Σ_1 of simple undecidability: you just need to simulate the system forward (as described in [AMP95] for the

two-dimensional case) and see whether the target point is reached within a finite number of discontinuities. Other approaches, such as "robust" (in the sense of insensitivity to small perturbations) realizations of transition systems by PCDs are currently investigated.

Beside the negative results, PCDs suggest an interesting model of computation which could theoretically[8] decide every statement in first-order arithmetics, i.e., solve every open problem in Number theory. This model which is more geometrical and topological in nature, may bring new insights on computability and synchronization and promote a new style of parallel programming. The art of PCD programming is to ensure that regions do not overlap and that the derivatives takes you where you want, usually using other variables as timers (or loop delimiters). For example, parallel sorting of n numbers can be implemented in linear time by PCDs using $3n/2 + 1$ dimensions.

As the reader might have noticed, the construction of lemma 4 involves an increase in the ordinality of the number of discontinuities in a trajectory. Higher levels of the hierarchy will lead to trajectories whose number of breakpints are higher ordinals. This fact connects our work with other investigations in higher recursion theory (see [S90]). Recently, O. Bournez [B97-a], showed that the ordinality of the number of intersections with boundaries in a PCD system is bounded by a function of the number of dimension, and hence gave a lower bound on the number of dimensions needed to realize each level in the hierarchy.[9] This shows, in particular, that 3-dimensional PCDs capture exactly the r.e. sets.

Acknowledgement This paper answers a question posed to us by Philippe Darondeau.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science* 138, 3–34, 1995.

[8] If we ignore physical limitations concerning the granularity of measurements.

[9] In fact, this holds only for PCDs whose limit point are always rational; otherwise, Bournez has shown [B97-b] that all the hierarchy can be realized in 5 dimensions.

[AKNS95] P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), *Hybrid Systems II*, LNCS 999, Springer, 1995.

[AMP95] A. Asarin, O. Maler and A. Pnueli, Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives, *Theoretical Computer Science* 138, 35-66, 1995.

[B97-a] O. Bournez, Some Bounds on the Computational Power of Piecewise Constant Derivative Systems, *Proc. ICALP'97* to appear.

[B97-b] O. Bournez, Personal Communication, 1997.

[R67] H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.

[S90] G.E. Sacks, Higher Recursion Theory, Springer, 1990.