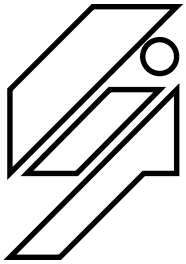
Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy.

Olivier Bournez



Laboratoire de l'Informatique du Parallélisme

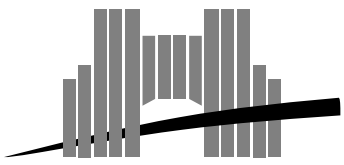
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy

Olivier Bournez

May 21, 1997

Research Report N^{O} 97-14



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) (0)4.72.72.80.00 Télécopieur : (+33) (0)4.72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy

Olivier Bournez

May 21, 1997

Abstract

We pursue the study of the computational power of Piecewise Constant Derivative (PCD) systems started in [5, 6]. PCD systems are dynamical systems defined by a piecewise constant differential equation and can be considered as computational machines working on a continuous space with a continuous time. We prove that the languages recognized by rational PCD systems in dimension $d = 2k + 3$ (respectively: $d = 2k + 4$), $k \geq 0$, in finite continuous time are precisely the languages of the $\omega^{k^{th}}$ (resp. $\omega^k + 1^{th}$) level of the hyper-arithmetical hierarchy. Hence the reachability problem for rational PCD systems of dimension $d = 2k + 3$ (resp. $d = 2k + 4$), $k \geq 1$, is hyper-arithmetical and is Σ_{ω^k}-complete (resp. Σ_{ω^k+1}-complete).

Keywords: Real computability, Continuous time computations, Dynamical systems, Hyper-arithmetical hierarchy

Résumé

Nous poursuivons l'étude de la puissance de calcul des systèmes à dérivée constante par morceaux débutée dans [5, 6]. Les systèmes PCD sont des systèmes dynamiques définis par une équation différentielle constante par morceaux. Ils peuvent être vus comme des modèles de calcul évoluant sur espace continu avec un temps continu. Nous prouvons que les langages reconnus par des systèmes PCD rationnels en dimension $d = 2k + 3$ (respectivement: $d = 2k + 4$), $k \geq 0$, en temps continu fini sont précisément les langages du $\omega^{k^{ieme}}$ (resp. $\omega^k + 1^{ieme}$) niveau de la hiérarchie hyper-arithmétique. Ainsi le problème de l'atteignabilité des systèmes PCD de dimension $d = 2k+3$ (resp. $d = 2k+4$), $k \geq 1$, est hyper-arithmétique et est Σ_{ω^k}-complet (resp. Σ_{ω^k+1}-complet).

Mots-clés: Calculabilité réelle, Calculs en temps continu, Hiérarchie hyper–arithmétique, Systèmes dynamiques.

1 Introduction

There has been recently an increasing interest in the fields of control theory and computer science about hybrid systems. A hybrid system is a system that combines discrete and continuous dynamics. Several models have been proposed in literature. Hybrid systems can be considered as computational machines [1, 2, 3, 7, 8]: they can be seen either as machines working on a continuous space with a discrete time or as machines working on a continuous space with a continuous time.

Several theoretical models of machines working on a continuous space with a discrete time are known: in [4], Blum, Shub and Smale introduce the real Turing machine. Many papers are devoted to this model: see [12] for an up-to-date survey. In [13], Meer introduces a restricted class of real Turing machines called the *linear machines*: Meer proves that $P \neq NP$ in this class of systems. In [10, 11], Koiran characterizes the boolean part of the languages recognized by linear machines as $P/poly$ in polynomial discrete time and as unbounded in exponential discrete time.

Simultaneously, in [1, 2, 7, 8], it is shown that several very simple dynamical systems can be considered as non trivial machines that work on a continuous space with a discrete time. In particular, in [1, 2, 3] the attention is focused on a very simple type of hybrid systems: Piecewise Constant Derivative Systems (PCD systems) are dynamical systems defined by a piecewise constant differential equation. It is shown that the reachability problem for PCD systems of dimension $d = 2$ is decidable and undecidable for dimensions $d \geq 3$ [1, 3] . In [7], the computational power of Piecewise Constant Derivative systems is characterized as $P/poly$ in polynomial discrete time, and as unbounded in exponential discrete time.

However, hybrid systems are very interesting models since they can be considered as natural computational machines working on a continuous space with a continuous time. The studies of this type of machines are only beginning. In [14], Moore proposes a recursion theory for computations on the reals in continuous time. Recently, Asarin and Maler [2] showed, using Zeno's paradox, that every set of the arithmetical hierarchy can be recognized in finite continuous time by a PCD system of finite dimension: every set of the arithmetical hierarchy in $\Sigma_k \cup \Pi_k$ can be recognized by a rational PCD system in dimension $5k + 1$. Unfortunately, no precise characterization of the sets recognizable by PCD systems is given in [2]. In [6], a precise characterization is given for the restricted class of purely rational PCD systems. However no answer is given about the general class of rational PCD systems.

We provide in this paper a full characterization of the computational power of rational PCD systems: we prove that every arithmetical set can be recognized in finite continuous time in dimension 5. Hence, in one sense, dimension 5 is universal for the arithmetical hierarchy. However, we prove in this paper that there does not exist a dimension d such that PCD systems of dimension d recognize every set of the hyper-arithmetical hierarchy: we prove that the languages recognized by rational PCD systems in dimension $d = 2k + 3$ (respectively: $d = 2k + 4$), $k \geq 0$, in finite continuous time are precisely the languages of the $\omega^{k^{th}}$ (resp. $\omega^k + 1^{th}$) level of the hyper-arithmetical hierarchy. In other words, the reachability problem for rational PCD systems of dimension $d = 2k + 3$ (resp. $d = 2k + 4$) is Σ_{ω^k}-complete (resp. Σ_{ω^k+1}-complete).

Section 2 is devoted to general definitions about Piecewise Constant Derivative Systems and about the hyper–arithmetical hierarchy. In section 3, we introduce Real Continuous Time (RCT) machines: we prove that RCT machines can recognize some hyper–arithmetical sets. In section 4, we show that RCT machines can be simulated by PCD systems and we deduce that PCD systems can also recognize some hyper–arithmetical sets. In section 5, we prove that the bounds given in section 4 are optimal: the languages recognized by rational PCD systems in dimension $d = 2k + 3$ (respectively: $d = 2k + 4$), $k \geq 0$ in finite continuous time are precisely the languages of the $\omega^{k^{th}}$ (resp. $\omega^k + 1^{th}$) level of the hyper-arithmetical hierarchy.

2 Definitions

2.1 PCD systems

A convex polyhedron of $\mathbb{R}^d$ is any finite intersection of open or closed half spaces of $\mathbb{R}^d$. A polyhedron of $\mathbb{R}^d$ is a finite union of convex polyhedral of $\mathbb{R}^d$. In particular, a polyhedron may be unbounded or flat. For $V \subset \mathbb{R}^d$, we denote by $\overline{V}$ the topological closure of V. We denote by d the distance of the maximum of $\mathbb{R}^d$.

Definition 2.1 (PCD System)

- *A* dynamical system *is a couple* $\mathcal{H} = (X, f)$ *where* $X = \mathbb{R}^d$ *and* f *is a function from* X *to* X. X *is called the* space *and* d *is called the* dimension *of* $\mathcal{H}$. *A* trajectory of $\mathcal{H}$ starting from x_0 *is a continuous solution to the differential equation* $\dot{x}_d = f(x)$, *with initial condition* x_0, *where* $\dot{x}_d$ *denotes the right derivative: that is to say* $\Phi : D \subset \mathbb{R}^+ \to X$ *where* D *is an interval of* $\mathbb{R}^+$ *containing* 0, $\Phi(0) = x_0$, *and* $\forall t \in D, \dot{\Phi}_d(t) = f(\Phi(t))$. *Trajectory* Φ *is said to* continue for ever *if* $D = \mathbb{R}^+$.

- *A* piecewise constant derivative (PCD) system *[2, 3] is a dynamical system* $\mathcal{H} = (X = \mathbb{R}^d, f)$ *where the range of* f *is a finite set* $C \subset X$, *such that for any* $c \in C$ *(*c *is called a* slope*)* $f^{-1}(c)$ *is a finite union of convex polyhedral sets (called* regions*).*

In other words a PCD system consists of partionning the space into convex polyhedral sets, called *regions*, and assigning a constant derivative c, called *slope* to all the points sharing the same region. The trajectories of such systems are broken lines with the breakpoints occuring on the boundaries of the regions [2]: see figure 1.

The *signature* of a trajectory is the sequence of the regions that are reached by the trajectory. The restriction of a trajectory $\Phi : D \to \mathbb{R}^d$ to interval $I \subset D$ is the restriction of function Φ to domain I.

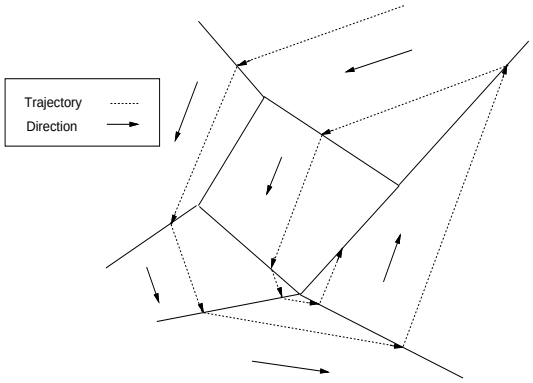


Figure 1: A PCD system in dimension 2.

In this paper we will deal only with rational PCD systems:

Definition 2.2

- *A PCD system is called* rational *if all the slopes as well as all the polyhedral regions can be described using only rational coefficients.*

- *A PCD system is called* purely rational, *if in addition, for all trajectory* Φ *starting from a rational point, every time* Φ *enters a region in some point* x, x *has rational coordinates.*

Some comments are in order: one must understand that a trajectory Φ can enter a region either by a discrete transition or by converging to a point of the region: see figure 3. Hence, the definition of purely rational PCD systems stipulates that every converging process converges towards a point with rational coordinates. We will see in theorem 5.1 that one can construct a rational PCD system of dimension 5 that is not purely rational.

We can say some words on the existence of trajectories in a PCD system: let $x_0 \in X$. We say that x_0 is *trajectory well-defined* if there exists a $\epsilon > 0$ such that $f(x) = f(x_0)$ for all $x \in [x_0, x_0 + \epsilon * f(x_0)]$. It is clear that, for any $x_0 \in X$, there exists a trajectory starting from x_0 iff x_0 is trajectory well-defined. Given a rational PCD system $\mathcal{H}$, one can effectively compute the set $NoEvolution(\mathcal{H})$ of the points of X that are not trajectory well-defined. See that a trajectory can continue for ever iff it does not reach $NoEvolution(\mathcal{H})$.

2.2 Computing with PCD systems

Let Σ be a finite alphabet with at least two letters. By renaming if necessary, we can assume without loss of generality that $\Sigma = \{1, 2, \ldots, n_\Sigma\}$.

We write Σ^* (respectively: Σ^ω) for the the set of the finite (resp. finite and infinite) words over alphabet Σ. We write ϵ for the empty word. If $w \in \Sigma^*$, we write $length(w)$ for the length of word w. We fix a recursive encoding of the integers over the words of Σ^*: for any integer $n \in \mathbb{N}$, we denote by $\overline{n}$ a word of Σ^* encoding integer n.

We describe now how to encode a word of Σ^* into a real of $[0,1]$. Denote by b_Σ the first power of 2 that is greater than $2n_\Sigma + 2$: $b_\Sigma = 2^{b'_\Sigma}$ for some $b'_\Sigma \in \mathbb{N}$.

Definition 2.3 (Encoding by $\mathcal{I}, \mathcal{J}$) *Let $\Sigma = \{1, 2, \ldots, n_\Sigma\}$ be the fixed finite alphabet.*

- *We denote by $\mathcal{J}$ the mapping from $\Sigma^\omega \to J$ that maps word $w = a_1 a_2 \ldots a_i \ldots$, with $a_1, a_2, \ldots \in \Sigma$, to real number*
$$\mathcal{J}(w) = \sum_{j=1}^{\infty} \frac{(2a_j)}{(b_\Sigma)^j}$$
- *We denote by $J \subset \mathbb{R}$ the range of $\mathcal{J}$.*
- *We denote by $\mathcal{I}$ the restriction of $\mathcal{J}$ to Σ^*.*
- *We denote by $I \subset \mathbb{R}$ the range of $\mathcal{I}$.*

PCD systems can be considered as machines recognizing some languages $L \subset \Sigma^*$ as follows:

Definition 2.4 (Computation [2])
- *Let $\mathcal{H} = (X, f)$ be a PCD system of dimension d. Let x^1, x^0 be two distinct points of $\mathbb{R}^d$. A computation of system $\hat{H} = (\mathbb{R}^d, f, \mathcal{I}, x^1, x^0)$ on entry $n \in \Sigma^*$ is a trajectory that can continue forever (defined on all $\mathbb{R}^+$) of $\mathcal{H} = (X, f)$ starting from $(\mathcal{I}(n), 0, \ldots, 0)$. The computation is accepting if the trajectory eventually reaches x^1, and refusing if it reaches x^0. It is assumed that the derivatives at x^1 and x^0 are zero.*
- *Language $L \subset \Sigma^*$ is semi-recognized by $\hat{H}$ if, for every $n \in \Sigma^*$, there is a computation on entry n and the computation is accepting iff $n \in L$. L is said to be (fully-)recognized by $\hat{H}$ when, in addition, this trajectory reaches x^0 iff $n \notin L$.*

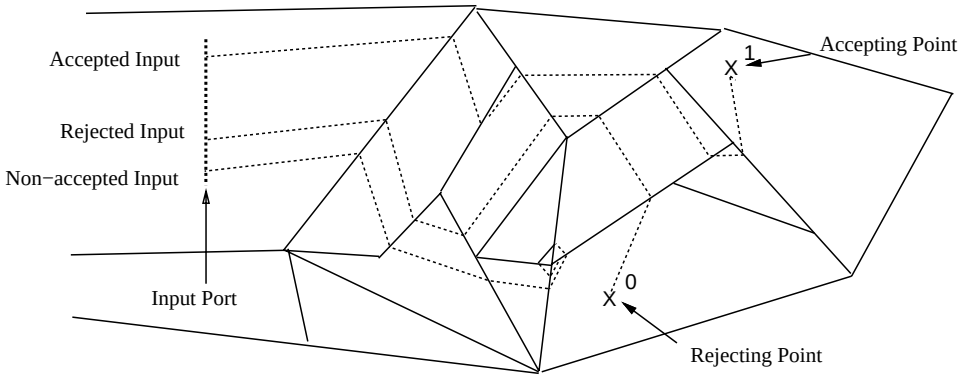


Figure 2: Some examples of computations by a PCD system.

2.3 Measuring the time on PCD systems

Definition 2.5 (Continuous and Discrete time) *Let $\Phi_n : \mathbb{R}^+ \to X$ be an accepting computation on entry $n \in \Sigma^*$.*

- *The continuous time $T_c(n)$ of the computation is $T = minimum\{t \in \mathbb{R}^+ / \Phi_n(t) = x^1\}$*

- *Let $T_n = \{t/\Phi_n$ reaches a boundary of a region at time $t\}$. It is easy to see that T_n is a well ordered set [6]. The discrete time $T_d(n)$ of the computation is defined as the order type of well ordered set T_n (= the ordinal corresponding to T_n).*

Note that Zeno's paradox appears: to a finite continuous time can correspond a transfinite discrete time: see figure 3.

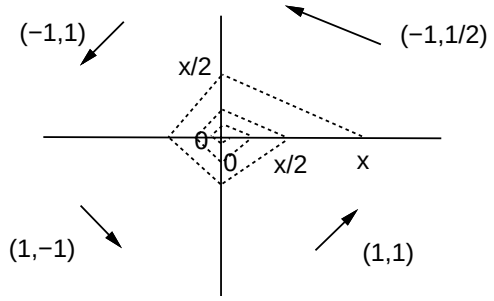


Figure 3: Zeno's paradox: at finite continuous time $5x = 2.5(x + x/2 + x/4 + \ldots)$ the trajectory is in $(0,0)$, but it takes a transfinite discrete time ω to reach this point.

2.4 Hyper-arithmetical hierarchy

2.4.1 Definition

For $y \in \mathbb{N}$, denote by M_y the y^{th} Turing machine, by $\phi_y : \Sigma^* \to \Sigma^*$ the function computed by M_y, and by $W_y \subset \Sigma^*$ the set of the words accepted by M_y. $y \in \mathbb{N}$ is said to be a *recursively enumerable index of* $X \subset \Sigma^*$ iff $X = W_y$. y is called an *recursive* index if in addition M_y halts on all inputs. Let $A \subset \Sigma^*$. Denote by an A exponent the relativizations to oracle A: M_y^A is the y^{th} Turing machine with oracle A, ϕ_y^A is the function computed by M_y^A, and W_y^A is the set of the words accepted by M_y^A. $y \in \mathbb{N}$ is said to be a *A-recursively enumerable index of* $X \subset \Sigma^*$ iff $X = W_y^A$. y is called an *A-recursive index* if in addition M_y^A halts on all inputs.

We follow the standard notations of [15, 16]: we fix a bijective recursive encoding of $\Sigma^* \times \Sigma^*$ into Σ^*: $< n, m >$ denotes a word of Σ^* encoding word $n \in \Sigma^*$ and word $m \in \Sigma^*$. For $X \subset \Sigma^*$, we denote $X' = \{\overline{x} | x \in \mathbb{N} \wedge \overline{x} \in W_x^X\}$. Let $A, B \subset \Sigma^*$. We write $A \leq_m B$ if there exists a recursive f such that, for all $w \in \Sigma^*$, $w \in A \Leftrightarrow f(w) \in B$. In that case, we say that $A \leq_m B$ via f. We write $A \leq_T B$ if A is B-recursive. Denote $A \equiv_T B$ iff $A \leq_T B$ and $B \leq_T A$.

We recall the following definitions:

Definition 2.6 (Constructive ordinals[16]) *We define by transfinite induction simultaneously $O \subset \mathbb{N}$, mapping $|\ |$ from 0 to a segment of the ordinal numbers and partial ordering $<_O$ on O.*

The ordinals in the range of $|\ |$ are called the constructive ordinals. *An ordinal α is said to have notation x iff $x \in O$ and $|x| = \alpha$.*

The transfinite induction is as follows:

- *Ordinal 0 receives notation 1: $1 \in O, |1| = 0$.*
- *Let γ be an ordinal. Assume that all the ordinals $< \gamma$ have received a notation, and assume that $<_o$ has been defined on these notations.*
  - *If $\gamma = \alpha + 1$ is a successor, γ receives notation 2^x, for all notation x of α: for all $x \in O$, if $|x| = \alpha$, then $2^x \in O, |2^x| = \gamma$ and $z <_0 2^x$ for all $z \in O$ with either $z = x$ or $z <_0 x$.*
  - *If γ is a limit, γ receives notation 3.5^y for all y such that $\{\phi_y(n)\}_{n=1}^{n=\infty}$ is an increasing sequence of notations of ordinals of limit γ: for all $y \in \mathbb{N}$, if $\{\phi_y(n)\}_{n=1}^{n=\infty}$ is a sequence of integers in*

O, if $\{|\phi_y(n)|\}_{n\equiv 1}^{n\equiv \infty}$ is an increasing sequence of ordinals with limit γ such that $\forall i \forall j\ i < j \Rightarrow \phi_y(i) <_0 \phi_y(j)$, then $3.5^y \in O, |3.5^y| = \gamma$ and $z <_0 3.5^y$ for all z for which there exists n such that $z <_0 \phi_y(n)$.

- *No other integer $y \in \mathbb{N}$ is in O.*

Denote $x \leq_o y$ if $x = y$ or $x <_0 y$.

Definition 2.7 *Let $X \subset \Sigma^*$. We define H as a mapping from O to the subsets of Σ^* by:*

- $H^X(1) = X$.
- $H^X(2^x) = (H^X(x))'$.
- $H^X(3.5^y) = \{< u, \overline{v} > | v \in O \ \wedge\ v <_0 3.5^y \ \wedge\ u \in H^X(v)\}$.

We note H for $H^\emptyset$. The following lemma is proved in [16]:

Lemma 2.1 (Spector [16]) *Let $X \subset \Sigma^*$. Let $x \in O, y \in O$ with $|x| = |y|$, then $H^X(x) \equiv_T H^X(y)$.*

As a consequence, for all constructive ordinal α, we can define the classes $\Sigma_\alpha^X, \Pi_\alpha^X, \Delta_\alpha^X$ unambiguously as follows:

Definition 2.8 (Hyper-arithmetical hierarchy) *Let $X \subset \Sigma^*$.*

- *For any constructive ordinal $1 \leq \alpha < \omega$, and for any y such that $\alpha = |2^y|$:*
  - *Σ_α^X is the class of the sets that are recursively enumerable in $H^X(y)$*
  - *Π_α^X is the class of the sets whose complement is in Σ_α^X.*
  - *$\Delta_\alpha^X = \Sigma_\alpha^X \cap \Pi_\alpha^X$.*
- *For any constructive ordinal $\alpha \geq \omega$ and for any y such that $|y| = \alpha$:*
  - *Σ_α^X is the class of the sets that are recursively enumerable in $H^X(y)$*
  - *Π_α^X is the class of the sets whose complement is in Σ_α^X.*
  - *$\Delta_\alpha^X = \Sigma_\alpha^X \cap \Pi_\alpha^X$.*

For all constructive ordinal α, we denote $\Sigma_\alpha, \Pi_\alpha, \Delta_\alpha$ for $\Sigma_\alpha^\emptyset, \Pi_\alpha^\emptyset, \Delta_\alpha^\emptyset$.

A set R is said to be hyper-arithmetical (respectively: X-hyper-arithmetical) if $R \in \Sigma_\beta$ (resp. $R \in \Sigma_\beta^X$) for some constructive ordinal β. R is said to be arithmetical (resp. X-arithmetical) iff in addition we have $\beta < \omega$. Check that the classes Σ_α for $1 \leq \alpha < \omega$ are precisely the classes of the arithmetical hierarchy defined in [2, 6].

One can easily prove:

Proposition 2.1 ([16]) *Let $A, B \subset \Sigma^*$ be some languages.*

- *For all $n \in \mathbb{N}$, for all $y \in O$ with $|y| = n$, the following conditions are equivalent*
  - *$B \in \Sigma_{n+1}^A$*
  - *B is recursively enumerable in $H^A(y)$*
  - *$B \leq_m H^A(2^y)$*
- *For all constructive ordinal $\beta \geq \omega$, for all $y \in O$ with $|y| = \beta$, the following conditions are equivalent*
  - *$B \in \Sigma_\beta^A$*
  - *B is recursively enumerable in $H^A(y)$*
  - *$B \leq_m H^A(2^y)$.*

We mention that the hyper-arithmetical hierarchy is strict: for all constructive ordinal $\alpha \geq \omega$ (respectively: $\alpha < \omega$), for all $y \in O, |y| = \alpha$, $H(2^y)$ (resp. $H(y)$) is in $\Sigma_\alpha - \cup_{\beta<\alpha}\Sigma_\beta$

We also mention that the hyper-arithmetical hierarchy can be related to the analytical hierarchy: see [16] for the definition of Δ^1_1.

Proposition 2.2 (Kleene [16]) *One has:*

$$\Delta^1_1 = \cup_{\beta \text{ constructive ordinal}} \Sigma_\beta$$

We will use the following two lemmas proved in [16]:

Lemma 2.2 ([16]) *There exists a recursive h such that for all $x, y \in O, x \leq_o y$, $H(x) \leq_m H(y)$ via $\phi_{h(x,y)}$.*

Lemma 2.3 ($+_o$) *There exists a recursive function $+_0$ of two variables such that for all $x, y \in O$,*

- $x +_0 y \in O$
- $|x +_0 y| = |x| + |y|$
- $y \neq 1 \Rightarrow x <_0 x +_o y$

3 Real Continuous Time machines and the hyper-arithmetical hierarchy

We introduce Real Continuous Time machines (RCT machines). We prove in this section that they can recognize every set of the arithmetical hierarchy and some sets of the hyper-arithmetical hierarchy. We will see in next section that PCD systems can simulate RCT machines.

3.1 RCT machines

3.1.1 First example

We present here an informal description of RCT machines. A formal definition will be given in next subsection.

We deal with machines that have a finite number d of real registers whose values can be any real of $[0, 1]$. These machines evolve according to a finite program made of assignments and of tests between the real registers. Any instruction I of these programs, that is to say any assignment or any test, has some associated real function $c_I : [0, 1]^d \to \mathbb{R}^+$ called *the cost of the instruction*. The execution of any instruction I takes a time equal to $c_I(x_1, \ldots, x_d)$, where $x_1, \ldots, x_d$ are the values of the real registers of the machine when the instruction is executed.

Write for example $x_1 := 2x_2$ $[x_3]$ for the instruction that replaces the value of real register x_1 by two times the value of real register x_2 in a time given by real register x_3: if this instruction is executed at time $t \in \mathbb{R}$, then the value of the first real register at time $t + x_3$ will be equal to two times the value of the second real register at time t, where x_3 is the value of the third real register at time t.

We want to use a special label "*limit**" to specify what to do when the time becomes Zeno.

Perhaps, the better is to consider a first example:

Algorithm 1 program "Hello world".

$x_1 := 1$ [1]
$x_2 := 1$ [1]
$x_3 := 0$ [1] /**set* $(x_1, x_2, x_3) = (1, 1, 0)$ *at time* 3.*/

while (true) **do**	/*$transform\ (x_1,x_2,x_3) = (1/2^n, 1/2^n, 1 + 1/2 + \ldots 1/2^{n-1})$ *at time* $3 + 3(1 + 1/2 + \ldots 1/2^{n-1})$ *for some* n, *to* $(x_1,x_2,x_3) = (1/2^{n+1}, 1/2^{n+1}, 1 + 1/2 + \ldots 1/2^n)$ *at time* $3 + 3(1 + 1/2 + \ldots 1/2^n)$*/
$x_3 := x_3 + x_1\ [x_2]$	
$x_1 := x_1/2\ [x_2]$	
$x_2 := x_2/2\ [x_2]$	
end while	
*limit**:	/**here, we have* $(x_1,x_2,x_3) = (0,0,1)$ *at time* 9. */
$x_1 := x_3\ [1]$	/**now set,* $(x_1,x_2,x_3) = (1,0,1)$ *at time* 10. */

Try to simulate the evolution of this program. At time 3, $(x_1,x_2,x_3) = (1,1,0)$ and the program is starting to execute the while loop. At time $3+3$ the program is starting to execute the loop for the second time. At time $3+3+3/2$ the program is executing the loop for the third time. At time $3+3+3/2+3/2^2+\ldots+3/2^{n-1}$, for all $n \in \mathbb{N}$, the program is executing the loop for the n^{th} time. And at time $9 = 3 + \sum_{j=0}^{\infty} 3/2^j$? ... The answer is the following: the program is executing the instruction labeled by *limit**. So this program is made such that, at time 9, the machine copies x_3 into x_1. Check that variables x_2 and x_1 tend to 0 and that variable x_3 tends to 1 during the execution of the infinite while loop. As a consequence, we consider that at time 9 the value of the first two real registers of the machine is 0 and that the value of the third real register is 1. Hence, the previous program is a program that always halts and that stops with $x_1 = 1, x_2 = 0, x_3 = 1$ at time 10.

In other words, we consider finite programs made of assignments and tests with a real cost, with a special label denoted by *limit**: label *limit** always denotes the instruction to do at a limit time, when the time becomes "Zeno": that is to say when a converging process happens in finite time.

3.1.2 Formal definitions

Now we give some formal definitions of the programs we consider. We will prove in section 4 that these programs can be simulated by PCD systems.

We consider programs made of instructions with a cost: the execution of an instruction I takes a time equal to the cost of instruction I.

Definition 3.1 (Instruction, Test)

- *An* assignment *in dimension d is a couple (f,c) where f, called the* operation, *is a partial mapping from $[0,1]^d$ to $[0,1]^d$ and c, called the* cost function, *is a partial mapping from $[0,1]^d$ to $\mathbb{R}^+$.*
- *A* test *in dimension d is a couple (R,c), where R is a partial relation over $[0,1]^d$, and c, called the* cost function, *is a partial mapping from $[0,1]^d$ to $\mathbb{R}^+$.*
- *An* instruction of dimension d *is either an assignment or a test of dimension d.*

For the simplicity of notations, we denote by "$x_i := g(x_1,\ldots,x_d)[c]$", the assignment (g',h') where, for all $x_1,\ldots,x_d \in [0,1]$, g' and h' are defined on $(x_1,\ldots,x_d)$ iff $g(x_1,\ldots,x_d) \in [0,1]$, and when g' and h' are defined on $(x_1,\ldots,x_d)$, then $g'(x_1,\ldots,x_d) = (x_1,\ldots,x_{i-1},g(x_1,\ldots,x_d),x_{i+1},\ldots,x_d)$ and $h'(x_1,\ldots,x_d) = c$. We denote by "$x_i := g(x_1,\ldots,x_d)$" the assignment $x_i := g(x_1,\ldots,x_d)\ [1]$. We denote by "$R?\ [c]$", where R is a relation, the test (R,c). We denote by "$R?$" the test $R?\ [1]$.

We will define below the set of the assignments and the set of the tests denoted by $Assgnmt_d$ and by $Test_d$ respectively that are admissible in dimension d.

A RCT machine of dimension d is a machine with d real registers that evolves according to its program. Its program is finite and is made of the assignments of $Assgnmt_d$ and of the tests of $Test_d$. The execution of any instruction takes a time equal to the cost of the instruction. Whenever the time becomes "Zeno" and

the variables converge, the machine enters a special limit state $limit^*$, and the execution goes on from this state. Formally:

Definition 3.2 (RCT machine) • *A* Real Continuous Time machine *(RCT machine) M, or a* RCT program *of dimension d, is a 6-uple $P = (Q, q_0, q_f^+, q_f^-, limit^*, \delta)$ where:*

- *Q is a finite set and $q_0, q_f^+, q_f^-, limit^* \in Q$.*
- *δ is a mapping from Q to $Q \times Q \times (Assgnmt_d \cup Test_d)$*

- *An* instantaneous description (ID) *of M is an element $(q, x_1, \ldots, x_d, t)$ of $Q \times [0,1]^d \times \mathbb{R}^+$. q is the* internal state, *t is the* time *and $x_1, \ldots, x_d$ are the* values of the real registers of M at time t.
- *Let $ID_1 = (q, x_1, \ldots, x_d, t)$ and $ID_2 = (q', x'_1, \ldots, x'_d, t')$ be two IDs of M. We write $ID_1 \vdash_d ID_2$ iff*
  - *either $\delta(q) = (q', q'', Assgnmt)$, with $Assgnmt = (f, c) \in Assgnmt_d$, and:*
    - *$(x_1, \ldots, x_d)$ is in the domain of function f and of function c.*
    - *$(x'_1, \ldots, x'_d) = f(x_1, \ldots, x_d)$.*
  - *or $\delta(q) = (q'', q''', Test)$, with $Test = (R, c) \in Test_d$, and:*
    - *$(x_1, \ldots, x_d)$ is in the domain of relation R and of function c.*
    - *$(x'_1, \ldots, x'_d) = (x_1, \ldots, x_d)$*
    - *$(q' = q''$ and $R(x_1, \ldots, x_d))$ or $(q' = q'''$ and $\neg R(x_1, \ldots, x_d))$.*
- *A computation of M starting from $(x_1, \ldots, x_d)$ is a sequence $(ID_i = (q^i, x_1^i, \ldots, x_d^i, t^i))_{i \leq I}$ of IDs of M, where I is an ordinal, such that:*
  - *$ID_0 = (q_0, x_1, \ldots, x_d, 0)$*
  - *For all $j \leq I$, if j is a successor then $ID_{j-1} \vdash_d ID_j$*
  - *For all $j \leq I$, if j is a limit point then*
    - *$\{t^{j'} | j' < j\}$ is a set bounded above by some real number.*
    - *for all $1 \leq i \leq d$, $lim_{j' \to j, j' < j} x_i^{j'}$ exists*
    - *$t^j = \sup\{t^{j'} | j' < j\}$*
    - *for all $1 \leq i \leq d$, $x_i^j = lim_{j' \to j, j' < j} x_i^{j'}$*
    - *$q^j = limit^*$*
- *The computation is accepting (respectively: rejecting) if there exists $j_0 \leq I$ (this implies that $t^{j_0} \in \mathbb{R}$ is finite) with $q^{j_0} = q_f^+$ (resp: $q^{j_0} = q_f^-$) and such that $\forall j < j_0, q^j \notin \{q_f^+, q_f^-\}$. In that case, $t^{j_0} \in \mathbb{R}$ is called the* (continuous) time of the computation. *If the computation is accepting, we say that M* maps $(x_1, \ldots, x_d)$ to $(x_1^{j_0}, \ldots, x_d^{j_0})$ in time t^{j_0}.
- *Program M can be considered as an instruction: the* assignment corresponding to the execution of program *M is the assignment (f', c') of dimension d where functions f' and c' are defined on $(x_1, \ldots, x_d) \in [0,1]^d$ iff there is an accepting computation starting from $(x_1, \ldots, x_d)$. When functions f' and c' are defined on $(x_1, \ldots, x_d)$ then $f'(x_1, \ldots, x_d) = (x'_1, \ldots, x'_d)$, $c'(x_1, \ldots, x_d) = t'$ iff M maps $(x_1, \ldots, x_d)$ to $(x'_1, \ldots, x'_d)$ in time t'.*

An instruction of dimension $d-1$ can be considered as an instruction of dimension d:

Definition 3.3 (Embedding instructions of dimension $d-1$ into dimension d) *Let $d \geq 2$ be an integer.*

- *Let (f, c) be an assignment in dimension $d-1$.*

  We still denote by (f, c) the assignment (f', c') of dimension d defined, for all $x_1, \ldots, x_d \in [0,1]$ by:

$$\begin{aligned} f'(x_1, \ldots, x_{d-1}, x_d) &= (f(x_1, \ldots, x_{d-1}), x_d) \\ c'(x_1, \ldots, x_{d-1}, x_d) &= c(x_1, \ldots, x_{d-1}) \end{aligned}$$

- *Let (R, c) be a test in dimension $d-1$.*

  We still denote by (R, c) the assignment (R', c') of dimension d defined, for all $x_1, \ldots, x_d \in [0,1]$ by:

$$\begin{array}{rcl} R(x_1, \ldots, x_{d-1}, x_d) & = & R(x_1, \ldots, x_{d-1}) \\ c'(x_1, \ldots, x_{d-1}, x_d) & = & c(x_1, \ldots, x_{d-1}) \end{array}$$

We define now the transformation $/x_{d+1}$ on instructions: this transformation is equivalent to making the change of variable x_i becomes x_i/x_{d+1} for all i:

Definition 3.4 (Transformation $/x_{d+1}$ on instructions) • *Let (f, c) be an assignment in dimension d. We write $(f/x_{d+1}, c/x_{d+1})$ for the assignment in dimension $d+1$ defined as follows:*

  - *f/x_{d+1} and $c/_{x_{d+1}}$ are defined on $(x_1, \ldots, x_{d+1})$ iff all the following conditions hold:*
    * $x_{d+1} > 0$
    * $(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \in [0,1]^d$
    * *function f and c are defined on value $(x_1/x_{d+1}, \ldots, x_d/x_{d+1})$*
  - *when f/x_{d+1} and $c/_{x_{d+1}}$ are defined on $(x_1, \ldots, x_d)$,*

$$\begin{array}{rcl} f/x_{d+1}(x_1, \ldots, x_{d+1}) & = & x_{d+1} f(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \\ c/x_{d+1}(x_1, \ldots, x_{d+1}) & = & x_{d+1} c(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \end{array}$$

- *Let (R, c) be a test in dimension d. We write $(R/x_{d+1}, c/x_{d+1})$ for the test in dimension $d+1$ defined as follows:*

  - *R/x_{d+1} and c/x_{d+1} are defined on $(x_1, \ldots, x_{d+1})$ iff all the following conditions hold:*
    * $x_{d+1} > 0$,
    * $(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \in [0,1]^d$
    * *R and c are defined on value $(x_1/x_{d+1}, \ldots, x_d/x_{d+1})$.*
  - *when R/x_{d+1} and c/x_{d+1} are defined on $(x_1, \ldots, x_{d+1})$,*

$$\begin{array}{rcl} R/x_{d+1}(x_1, \ldots, x_{d+1}) & = & R(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \\ c/x_{d+1}(x_1, \ldots, x_{d+1}) & = & x_{d+1} c(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \end{array}$$

Take an example: consider instruction I defined as $x_1 := x_1 + \lambda\ [1]$ where $\lambda \in \mathbb{Q}$. When $x_3 > 0$ and $x_1/x_3 \in\ [0,1]$, I/x_3 is equivalent to instruction $x_1 := x_1 + \lambda x_3[x_3]$. When $x_3 = 0$ or $x_1/x_3 > 1$ then no evolution is possible.

We are ready to define the admissible operations in dim d: this is done inductively.

Definition 3.5 (Admissible operations in dim d) *We define inductively the set of the assignments denoted by $Assgnmt_d$ (respectively: the set of the tests denoted by $Test_d$) that are admissible in dimension d:*

For all d, for all $i, j, k \in \{1, 2, \ldots, d\}$, for all $\lambda \in \mathbb{Q}$, for all $\lambda^+ \in \mathbb{Q}^+$, for all $\# \in \{>, \geq, <, \leq, =, \neq\}$:

- *"Linear machines instructions"*
  - "$x_i := x_i + x_k\ [1]$" $\in Assgnmt_d$
  - "$x_i := x_j\ [1]$" $\in Assgnmt_d$.
  - "$x_i := \lambda^+ x_i\ [1]$" $\in Assgnmt_d$.
  - "$x_i := \lambda\ [1]$" $\in Assgnmt_d$.
  - "$x_i := x_i + \lambda\ [1]$" $\in Assgnmt_d$.
  - "$x_i \# \lambda^+ ?\ [1]$" $\in Test_d$.
- *"Special instructions"*

- "$x_d := x_d/2\ [x_d]$" $\in Assgnmt_d$, *if* $d > 2$.
- "$x_d := 2x_d\ [x_d]$" $\in Assgnmt_d$, *if* $d > 2$.
- "$x_d := x_d + \lambda x_k\ [x_k]$" $\in Assgnmt_d$, *if* $2 < k < d$

- *"Subprograms"*
  - *If P is a program of dimension d, then $(f, c+1) \in Assgnmt_d$, where (f, c) is the assignment corresponding to the execution of P.*

- "$Assgnmt_{d-1} \subset Assgnmt_d$", "$Test_{d-1} \subset Test_d$"
  - *If $(f, c) \in Assgnmt_{d-1}$ then $(f, c) \in Assgnmt_d$.*
  - *If $(R, c) \in Test_{d-1}$ then $(R, c) \in Test_d$.*

- *"Zeno instructions"*
  - *If $(f, c) \in Assgnmt_{d-1}$ and $d > 2$ then $(f/x_d, c/x_d) \in Assgnmt_d$.*
  - *If $(R, c) \in Test_{d-1}$ and $d > 2$ then $(R/x_d, c/x_d) \in Test_d$.*

RCT machines can be considered as machines recognizing some languages $L \subset \Sigma^*$ as follows:

Definition 3.6 *Let Σ be the fixed finite alphabet.*

Language $L \subset \Sigma^*$ is semi-recognized by RCT machine M *if, for all $n \in \Sigma^*$, M has an accepting computation starting from $(\mathcal{I}(n), 0, \ldots, 0)$ iff $n \in L$.*

L is fully-recognized *if in addition, for all $n \in \Sigma^*$, there is a rejecting computation starting from $(\mathcal{I}(n), 0, \ldots, 0)$ iff $n \notin L$.*

We will write the RCT programs in a high level programming language style using all the usual flow control instructions (while, if, for, goto) as in algorithm 1.

3.2 RCT machines can simulate Turing machines

We show in this subsection that one can simulate any Turing machine by a RCT machine. This is nothing but a restatement of [10].

3.2.1 Two stacks pushdown automata can simulate Turing machines

It is well known that Turing machines are equivalent to two stacks pushdown automata (2PDA)[9]: the two stacks correspond to the content of the tape on the right and on the left respectively of the head of the Turing machine.

We go here into the precise details of what we call a 2PDA or an ω-2PDA. A 2PDA is equivalent to a Turing machine, and an ω-2PDA is equivalent to a Turing machine with a semi-infinite tape.

Write Σ' for $\Sigma \cup \{\epsilon\}$. If $w \in \Sigma^\omega$, write w^+ for the element of Σ' equal to the first letter of w if $w \neq \epsilon$ and equal to ϵ if $w = \epsilon$. If $w \neq \epsilon$, denote w^- for the word such that $w = w^+ w^-$. If $i \in \{1, 2\}$, denote $\tilde{i} = 3 - i$.

Definition 3.7 (2PDA)
- *A* two stacks pushdown automaton *(2PDA) (respectively: an ω-2PDA), is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set, $q_0 \in Q$ is the initial state, $F \subset Q$ is the set of the final states, δ is a mapping from $Q \times \Sigma' \times \Sigma'$ to $Q \times Q \times InstructionSet$, where $InstructionSet$ is the following finite set of symbols: $InstructionSet = \{Push^i(a), Pop^i(a), Top^i(b)? | i \in \{1, 2\}, a \in \Sigma, b \in \Sigma'\}$*

- *An Instantaneous Description (ID) of a 2PDA (resp. of an ω-2PDA) is an element (q, w_1, w_2) of $Q \times \Sigma^* \times \Sigma^*$ (resp. of $Q \times \Sigma^\omega \times \Sigma^*$). q is the internal state, w_1,w_2 are the contents of the first and the second stack respectively. The ID is accepted iff $q \in F$.*

- *The relation $\vdash_d$ between IDs is defined as follows:*

$$(q, w_1, w_2) \vdash_d (q', w_1', w_2')$$

$$iff \left\{ \begin{array}{l} \delta(q, w_1^+, w_2^+) = (q'', q''', instr) \\ \quad and \left\{ \begin{array}{ll} & (instr = Push^i(a), q' = q'', \\ & w_i' = aw_i, w_{\bar{i}}' = w_{\bar{i}}) \\ or & (instr = Pop^i(a), q' = q'', w_i^+ = a, \\ & w_i' = (w_i)^-, w_{\bar{i}}' = w_{\bar{i}}) \\ or & (instr = Top^i(b)?, w_1' = w_1, w_2' = w_2, \\ & \quad and \left\{ \begin{array}{ll} & (q' = q'' \ and \ w_i^+ = b) \\ or & (q' = q''' \ and \ w_i^+ \neq b)) \end{array} \right. \end{array} \right. \end{array} \right.$$

- *Let $\vdash_d^*$ be the transitive closure of $\vdash_d$. An input $w \in \Sigma^*$ (respectively $w \in \Sigma^\omega$) is accepted by a 2PDA (resp. by an ω-2PDA) iff there exists an accepted ID ID_{acc} such that $(q_0, w, \epsilon) \vdash_d^* ID_{acc}$.*
- *We say that a 2PDA (resp.an ω-2PDA) maps (w_1, w_2) to (w_1', w_2'), where $w_1, w_2, w_1', w_2' \in \Sigma^*$ (resp. $w_1, w_1' \in \Sigma^\omega, w_2, w_2' \in \Sigma^*$) , iff $(q_0, w_1, w_2) \vdash_d^* (q, w_1', w_2')$ for some $q \in F$.*

3.2.2 RCT machines can simulate Two stacks pushdown automata

We show now that RCT machines can simulate 2PDAs using only the linear machine instructions: this is a restatement of [10].

Lemma 3.1 *One can simulate any two stacks pushdown automaton (respectively any ω-two stacks pushdown automaton) M by a RCT machine M' of dimension 2 whose program is only made of the linear machine instructions.*

Proof: One build a RCT machine M' that simulates M: when the stacks of M are words w_1, w_2, the registers of M' are $x_1 = \mathcal{J}(w_1)$ and $x_2 = \mathcal{J}(w_2)$. The program of M' is obtained by taking the program of M, and by replacing one after the other the 2PDA instructions of M by some linear machine instructions using the correspondence of figure 4.

□

Theorem 3.1 *Let S be a discrete language.*

- *Assume that S is recursively enumerable. Then S is semi-recognized by a RCT machine of dimension 2*
- *Assume that S is recursive. Then S is fully-recognized by a RCT machine of dimension 2.*

Proof: Immediate from lemma 3.1 and from the fact that any Turing machine can be simulated by a two stacks pushdown automaton [9].

□

Convention 3.1 *We use the following convention:*

$$\left[\begin{array}{l} (w_1, w_2) \\ \mapsto (w_1', w_2') \\ \textbf{where} \quad \textit{"conditions"} \end{array} \right]$$

denotes any RCT program M' that, for all w_1, w_2 verifying "conditions", maps real registers $x_1 = \mathcal{J}(w_1)$, $x_2 = \mathcal{J}(w_2)$ to $x_1 = \mathcal{J}(w_1'), x_2 = \mathcal{J}(w_2')$: to obtain M', consider any ω-2PDA M such that, for all $w_1 \in \Sigma^\omega, w_2 \in \Sigma^*$ verifying "conditions", M maps (w_1, w_2) to (w_1', w_2'): recall that 2PDAs are equivalent to Turing machines. Now apply the transformation of the proof of lemma 3.1 on M to get RCT machine M'.

As an example, $\left[\begin{array}{l} (w, w') \\ \mapsto (ww, \epsilon) \\ \textbf{where} \quad w, w' \in \Sigma^*, w = w' \end{array} \right]$ is any RCT program that, for all $w, w' \in \Sigma^*$ such that $w = w'$, maps $(\mathcal{J}(w), \mathcal{J}(w'))$ to $(\mathcal{J}(ww), 0)$.

Two stacks automata instruction	RCT machine instructions
$Push^i(j), i \in \{1,2\}, j \in \Sigma$	$x_i := x_i/b_\Sigma + (2*j)/b_\Sigma$ [1]
$Pop^i(j), i \in \{1,2\}, j \in \Sigma$	$x_1 := b_\Sigma * (x_1 - (2*j)/b_\Sigma)$ [1]
$Top^i(\epsilon)?, i \in \{1,2\}$	$x_i = 0?$ [1]
$Top^i(j)?, i \in \{1,2\}, j \in \Sigma$	$((x_i \geq (2*j)/b_\Sigma)?$ [1] and $(x_i \leq (2*j+1)/b_\Sigma)?$ [1])

Figure 4: Correspondence between 2PDA instructions and RCT instructions. The RCT instructions corresponding to the 2PDA instructions $Push^i(j), Pop^i(j), Top^i(j)?$ will be still denoted by $Push^i(j)$, $Pop^i(j)$, $Top^i(j)?$.

3.3 RCT machines and the arithmetical hierarchy

3.3.1 Speedup properties of RCT machines

In definition 3.4, we defined the transformation $/x_{d+1}$ on instructions: the transformation $/x_{d+1}$ on a RCT program P is obtained by transforming instruction by instruction the instructions of P:

Definition 3.8 (Transformation $/x_{d+1}$ on RCT programs) *Let P be a RCT program of dimension d: $P = (Q, q_0, q_f^+, q_f^-, limit^*, \delta)$.*

We denote by P/x_{d+1} the RCT program of dimension $d+1$ defined by $P/x_{d+1} = (Q, q_0, q_f^+, q_f^-, limit^*, \delta')$ where for all q, q', q'' and $Instr \in Test_d \cup Assgnmt_d$, $\delta(q) = (q', q'', Instr) \Leftrightarrow \delta'(q) = (q', q'', Instr/x_{d+1})$

We prove:

Lemma 3.2 (Speedup lemma) *Let P be a RCT program of dimension d.*

For all $\lambda \in (0,1]$, for all $x_1, x_2, \ldots, x_d \in [0,1]$, P/x_{d+1} started with real registers $(\lambda x_1, \lambda x_2, \ldots, \lambda x_d, \lambda)$ simulates the evolution of P on $(x_1, \ldots, x_d)$ but the simulation of P by P/x_{d+1} goes $1/\lambda$ times faster than P.

Proof:

Let $\lambda \in (0,1]$ and let $x_1, \ldots, x_d \in [0,1]$ be fixed.

Denote by $(q^j, x_1^j, \ldots, x_d^j, t^j)_{j \in J}$ the computation of P starting from $(x_1, \ldots, x_d)$.

Denote by $(q'^j, x_1'^{j'}, \ldots, x_d'^{j'}, x_{d+1}'^{j'}, t'^{j'})_{j' \in J'}$ the computation of P' starting from $(\lambda x_1, \ldots, \lambda x_d, \lambda)$. It is easy to prove by transfinite induction that, for all $j \in J$, one has $j \in J'$, $q^j = q'^j$, $x_{d+1}'^j = \lambda$, $x_i'^j = \lambda x_i^j$ for all $1 \leq i \leq d$, and $t'^j = \lambda t^j$.

□

We get immediately:

Theorem 3.2 *Assume that S is semi-recognized (respectively: fully-recognized) by a program P in dimension d in time T. For any $k \in \mathbb{N}^+$, S is semi-recognized (resp. fully-recognized) in dimension $d+1$ in time $2+T/k$.*

Proof:

Assume that S is semi-recognized by P. S is semi-recognized (resp. fully-recognized) in dimension $d+1$ in time $2+T/k$ by the following program P':

Algorithm 2 Program P'

$x_1 := x_1/k$ [1]
$x_{d+1} := 1/k$ [1]
P/x_{d+1}

□

3.3.2 From semi-recognition to recognition

We see now that one can transform a program that semi-recognizes a set S in dimension d to a program that fully-recognizes S in finite time in dimension $d+1$.

We need first some definitions: a clocked program is a program where some instructions are marked and where the execution of the marked instructions can be used as the tops of a clock: there must exists an upper bound Δ for the time between two successive tops, and any bounded time interval must contain a finite number of tops:

Definition 3.9

- *A* clocked program *is any program* $P = (Q, q_0, q_f^+, q_f^-, limit^*, \delta)$ *such that:*
  - *some instructions of* P *are marked: there exists some* $Q_0 \subset Q$.
  - *there exists* $\Delta \in \mathbb{R}^+$, *called the* time period of P, *such that any computation of* P *executes a finite number* $n_I \geq 1$ *of marked instructions on any time interval* I *of width* Δ: *for all computation* $C = (q^j, x_1^j, \ldots, x_d^j, t^j)_{j \in J}$ *of* P, *for all* $t \in \mathbb{R}^+$, *if there is some* $j \in J$ *with* $t^j \geq t + \Delta$, *then the cardinality of the set* $\{j' | j' \in J, q^{j'} \in Q_0, t^{j'} \in [t, t+\Delta]\}$ *is finite and greater than* 1.
- *An* ω-clocked *program is a clocked program such that a computation of* P *executes a finite number of marked instructions iff the computation is accepting.*
- *If* P *is a clock program, and* R *is a program, we write* $P * R$ *for the program that one gets by inserting in* P *a copy of program* R *at each marked instruction of* P

As an example, take any 2PDA M that semi-recognizes a set $L \subset \Sigma^*$. Assume that M never enters a "reject" state but loops for ever on an input $w \in \Sigma^*$ with $w \notin L$. Then the RCT program M' of dimension 2 given by lemma 3.1 that simulates M can be ω-clocked: mark all the instructions of M' and take $\Delta = 1$ as time period.

We use the following convention: we assume any program given by lemma 3.1 (and therefore any program given by the notation $\left[\begin{array}{l}(w_1, w_2) \\ \mapsto (w_1', w_2') \\ \textbf{where} \ \text{conditions}\end{array}\right]$) marked with all the instructions marked. Moreover, when P is marked, we consider that P/x_{d+1} is marked, where the marked instructions of P/x_{d+1} are the instructions corresponding to the marked instructions of P. In particular, if R is a program, and P is a marked program then $(P/x_{d+1}) * R$ is the program that one gets by transforming all the instructions of P by transformation $/x_{d+1}$ and by inserting a copy of R at each instruction corresponding to a marked instruction of P.

Let $d \geq 2$ be an integer. We will use the following program:

Algorithm 3 Program $Div2^{d+1}$

$x_1 := x_1/2 \ [x_{d+1}]$
$x_2 := x_2/2 \ [x_{d+1}]$
$\ldots$
$x_{d-1} := x_{d-1}/2 \ [x_{d+1}]$
$x_d := x_d/2 \ [x_{d+1}]$
$x_{d+1} := x_{d+1}/2 \ [x_{d+1}]$

And the following program:

Algorithm 4 Program $Mul2^{d+1}$

$x_1 := 2x_1 \ [x_{d+1}]$
$x_2 := 2x_2 \ [x_{d+1}]$
$\ldots$
$x_{d-1} := 2x_{d-1} \ [x_{d+1}]$
$x_d := 2x_d \ [x_{d+1}]$

We claim:

Theorem 3.3 *Assume that S is semi-recognized by an ω-clocked program in dimension d. Then:*

- *There exists a program of dimension $d+1$ that fully recognizes S.*
- *Moreover, for all $k \in \mathbb{N}^+$, there exists a program of dimension $d+1$ that fully recognizes S in time $2 + 1/k$.*

Proof:
Assume that S is semi-recognized by P. Let $0 < \lambda \leq 1$ be some rational constant.
Consider the following program P'_λ

Algorithm 5 P'_λ

$x_1 := \lambda x_1$ [1]
$x_{d+1} := \lambda$ [1]
$(P/x_{d+1}) * (Div2^{d+1})$
*limit**: Reject

It is sufficient to take P'_1 to prove the first assertion and to take $P'_{1/(2k(\Delta+d+1))}$ to prove the second assertion, using lemma 3.3 proved below.

□

Here is the trick:

Lemma 3.3 (Super-speedup lemma) *Let P be an ω-clocked RCT program of dimension d of time period Δ. For all $\lambda \in (0,1]$, for all $x_1, x_2, \ldots, x_d \in [0,1]$, $(P/x_{d+1}) * Div2^{d+1}$ started with real registers $(\lambda x_1, \lambda x_2, \ldots, \lambda x_d, \lambda)$ simulates the evolution of P on $(x_1, \ldots, x_k)$ but the whole simulation of P by P/x_{d+1} is made in a finite bounded time upper bounded by $2\lambda(\Delta + d + 1)$.*

Moreover, whenever P accepts, $(P/x_{d+1}) * Div2^{d+1}$ accepts. Whenever P does not accept, $(P/x_{d+1}) * Div2^{d+1}$ converges to its limit state with all its real registers set to 0.

Proof: Let $\lambda \in (0,1]$ and $x_1, \ldots, x_d \in [0,1]$ be fixed.

Denote by $(q^j, x_1^j, \ldots, x_d^j, t^j)_{j \in J}$ the computation of P starting from $(x_1, \ldots, x_d)$. Let $Q_0 \subset Q$ gives the marked instructions of P.

Denote by $(q'^{j'}, x_1'^{j'}, \ldots, x_d'^{j'}, t'^{j'})_{j' \in J'}$ the computation of $(P/x_{d+1}) * Div2^{d+1}$ starting from $(\lambda x_1, \ldots, \lambda x_d, \lambda)$.

Denote by $j_1 < j_2 < \ldots \in J$ the sequence of the indexes corresponding to the execution of the marked instructions of P: for all $j \in J$, either $q^j \in Q_0$ and $j = j_k$ for some k or $q^j \notin Q_0$.

For $j \in J$, let n_j denote the number of marked instructions of P executed between time 0 and time t^j: n_j is the cardinality of set $\{k | t^{j_k} < t^j\}$.

It is easy to prove by transfinite induction on $j \in J$ that, for all $j \in J'$, one has $j \in J$, $q'^{j+n_j(d+1)} = q^j, x_i'^{j+n_j(d+1)} = \lambda x_i^j / 2^{n_j}$ for all $1 \leq i \leq d$, $x_{d+1}'^{j+n_j(d+1)} = \lambda/2^{n_j}$, $t'^{j+n_j(d+1)} = \lambda \sum_{k=1}^{n_j} (t^{j_k} - t^{j_{k-1}} + d + 1)/2^{k-1} + \lambda(t^j - t^{j_{n_j}})/2^{n_j}$ with $t^0 = 0$, and that for all k and $l \leq d+1$, $q'^{t^{j_k}+(k-1)(d+1)+l}$ corresponds to an instruction of program $Div2^{d+1}$.

This means that $(P/x_{d+1}) * Div2^{d+1}$ simulates P: if P accepts then $(P/x_{d+1}) * Div2^{d+1}$ accepts: $q^{j_0} = q_f^+$ for some $j_0 \in J$ implies $q'^{j_0+n_{j_0}(d+1)} = q_f^+$. For all $k \in \mathbb{N}$, we have $t^{j_k} - t^{j_{k-1}} \leq \Delta$. As a consequence, for all $j \in J$, $t^j \leq 2\lambda(\Delta + d + 1)$. Hence, $(P/x_{d+1}) * Div2^{d+1}$ accepts at some finite time bounded above by $2\lambda(\Delta + d + 1)$.

If P does not accept its input, since P is assumed to be ω-clocked, a non finite number of $Div2^{d+1}$ are executed. As a consequence, for all $1 \leq i \leq d+1$, the sequence $(x_i^j)_{j \in J}$ converge to 0. One has have

$sup_{j\in J} t^j \leq 2\lambda(\Delta+d+1)$. That means that $(P/x_{d+1}) * Div2^{d+1}$ reaches the ID $(limit^*, 0, \ldots, 0, t^*)$ at finite time $t^* = sup_{j\in J} t'^j$, with $t^* \leq 2\lambda(\Delta+d+1)$.

□

3.3.3 Recognizing arithmetical sets

The following lemma will be used in the proof of lemma 3.5:

Lemma 3.4 *There exists an injective mapping $\mathcal{E}$ from I to $(1/2, 1]$ such that, for any $d \geq 3$,*

- *There exists a RCT program Enc_d of dimension d that, for all $y_1 \in I$, maps $(y_1, ., \ldots, ., .)$ to $(\mathcal{E}(y_1)y_1, ., \ldots, \mathcal{E}(y_1))$.*
- *There exists a RCT program Dec_d of dimension d that maps $(., \ldots, ., \mathcal{E}(y_1))$ to $(y_1, ., \ldots, .)$ for all $y_1 \in I$.*

We say that a RCT machine M of dimension d maps $(\alpha_1, \ldots, \alpha_d)$ to $(\beta_1, \ldots, \beta_d)$, where for all $1 \leq i \leq d$, either $\alpha_i \in [0, 1]$ or α_i is the symbol ., and either $\beta_i \in [0, 1]$ or β_i is the symbol ., iff for all $(x_1, \ldots, x_d)$ with $x_i = \alpha_i$ for all i such that $\alpha_i \neq .$, M maps $(x_1, \ldots, x_d)$ to some $(x'_1, \ldots, x'_d)$ with $x'_i = \beta_i$ for all i such that $\beta_i \neq .$.

Proof: Let # be a letter of Σ. Denote by $number : \Sigma^* \to \mathbb{N}$ the function that maps any word $w \in \Sigma^*$ onto its number in some fixed recursive enumeration of the words of Σ^*. For all $n \in \mathbb{N}$, denote by w_n the n^{th} word of Σ^*: $number(w_n) = n$.

For all $w \in \Sigma^*$, we define $\mathcal{E}(\mathcal{I}(w))$ as the unique point which is simultaneously in interval $(1/2, 1]$ and in the set $\{2^k/3^{number(w)} | k \in \mathbb{Z}\}$.

It is sufficient to consider program Enc_d as the following RCT program:

Algorithm 6 Enc_d

$\left[\begin{array}{l} (w, w') \\ \mapsto (w, \#^{number(w)}) \\ \textbf{where} \quad w, w' \in \Sigma^* \end{array}\right]$

/*Set x_2 to $\mathcal{I}(\#^{number(w)})$*/

$x_d := 1$

while $(x_2 \neq 0?)$ **do** /*Do $number(w)$ times $x_1 := x_1/3, x_d = x_d/3$ */

  $x_2 := b_\Sigma * (x_2 - (2 * \#)/b_\Sigma)$

  $x_1 := x_1/3$

  $x_d := x_d/3$

end while

/*Now, while $x_d \notin (1/2, 1]$ multiply x_1 and x_d by 2*/

while $(x_d \leq 1/2?)$ **do**

  $x_1 := 2x_1$

  $x_d := 2x_d$

end while

And program Dec_d as:

Algorithm 7 Dec_d

$x_2 := 0$ /*Do $n := 0$: set $x_2 = \mathcal{I}(\#^0)$*/

while $(x_d \neq 1?)$ **do** /*While ($x_d \neq 2^k/3^n$ for some k) do*/

  $x_2 := x_2/b_\Sigma + \#/b_\Sigma$ /*Do $n := n + 1$: transform $x_2 = \mathcal{I}(\#^n)$ into $x_2 = \mathcal{I}(\#^{n+1})$*/

  $x_d := 3x_d$

while $(x_d > 1?)$ **do**
$x_d := x_d/2$
end while
end while
$\left[\begin{array}{l}(w, \#^n) \\ \mapsto (w_n, \epsilon) \\ \textbf{where} \quad w \in \Sigma^*, n \in \mathbb{N}\end{array}\right.$ /*Put $x_1 = \mathcal{I}(w_n)$.*/

□

We need to improve theorem 3.3: we show that any language that is semi-recognized in dimension d can be fully-recognized in dimension $d+1$ by a clocked program that returns its input when it accepts.

Lemma 3.5 *Let S be a discrete language. Assume that S is semi-recognized by an ω-clocked program P in dimension d. Then*
S is fully-recognized in dimension $d+1$ by some clocked program $\tilde{P}$: for all $w \in \Sigma^*$

- *if $w \notin S$ then $\tilde{P}$ rejects input $\mathcal{J}(w)$ and stops with all its real registers set to 0.*
- *if $w \in S$ then $\tilde{P}$ accepts input $\mathcal{J}(w)$ and stops with its first real register set to $\mathcal{J}(w)$ and all its other real registers set to 0.*

Proof:
Denote by Enc_{d+1} and Dec_{d+1} the programs of lemma 3.4, and by $\mathcal{E} : I \to (1/2, 1]$ the function of lemma 3.4.
Take $\tilde{P}$ defined as follows:

Algorithm 8 Program $\tilde{P}$

Enc_{d+1}	/*Maps $x_1 = y_1$ to $x_1 = \mathcal{E}(y_1)y_1, x_{d+1} = \mathcal{E}(y_1)$*/
$(P/x_{d+1}) * (Div2^{x_{d+1}})$	/*Simulate P on input y_1*/
while $(x_{d+1} \leq 1/2?\ [1])$ **do**	
$x_{d+1} := 2x_{d+1}\ [1]$	
end while	/*If it stops then undo all the divisions by 2 done on variable x_{d+1} during the simulation. At the end of the while loop we have $x_{d+1} = \mathcal{E}(y_1)$.*/
Dec_{d+1}	/*Map $x_{d+1} = \mathcal{E}(y_1)$ to $x_1 = y_1$.*/
$x_{d+1} := 0\ [1]$	
$x_2 := 0\ [1]$	
Accept	/*And accept.*/
*limit**:	
Reject	/*Else reject.*/

Let $x_1, \ldots, x_d, x_{d+1} \in [0, 1]$ be fixed. If $(x_1, \ldots, x_d)$ is not accepted by P, then from lemma 3.3, $(x_1, \ldots, x_d, x_{d+1})$ is rejected by this program, and all the registers are set to 0.
Now, assume $(x_1, \ldots, x_d)$ is accepted: by the proof of lemma 3.3, RCT program $(P/x_{d+1}) * (Div2^{x_{d+1}})$ stops with its $d+1^{th}$ real register equal to $\mathcal{E}(x_1)/2^{n_{j_0}}$, where $n_{j_0} \in \mathbb{N}$ is the number of marked instructions of the accepting computation of P starting from $(x_1, \ldots, x_d)$. It is easy to see that the while loop will be executed n_{j_0} times and that the end of the while loop the content of the $d+1^{th}$ real register will be set to $\mathcal{E}(x_1)$. Hence, after the Dec_{d+1} program, the first real register of $\tilde{P}$ will return to value x_1.
By lemma 3.3, program $(P/x_{d+1}) * (Div2^{x_{d+1}})$ is always executed in a bounded time. As a consequence, $\tilde{P}$ can be clocked: mark all the instructions but those of $(P/x_{d+1}) * (Div2^{x_{d+1}})$. Take as time period of $\tilde{P}$ the maximum of 1 and of the time needed to execute program $(P/x_{d+1}) * (Div2^{x_{d+1}})$.

□

As a consequence, we get:

Lemma 3.6 *Assume that B is a discrete language such that all the languages of Σ_1^B are semi-recognized by some ω-clocked RCT program in dimension $d' \geq 2$.*

Let S be a discrete language with $S \in \Sigma_k^B$, $k \in \mathbb{N}, k \geq 1$. Then S is semi-recognized by an ω-clocked RCT program in dimension $d' + k - 1$.

Proof: We prove the assertion by induction over k.

Case $k = 1$ is true by hypothesis.

Assume $k \geq 2$ and the hypothesis at rank $k-1$. Let $S \in \Sigma_k^B$. There exists $S' \in \Sigma_{k-1}^B$ such that $x \in S \Leftrightarrow \exists n \in \mathbb{N} \ \ < \overline{n}, x > \notin S'$: see [16]. By induction hypothesis S' is semi-recognized in dimension $k + d' - 2$ by an ω-clocked RCT program P_{k-1}. Let $\tilde{P_{k-1}}$ be the marked program that one gets by applying lemma 3.5 on program P_{k-1}.

S is semi-recognized by the following RCT program P_k:

Algorithm 9 Program P_k

$\left[\begin{array}{l} (w, w') \\ \mapsto (< \overline{0}, w >, w') \\ \textbf{where} \ \ w, w' \in \Sigma^* \end{array} \right]$
while ($\tilde{P_{k-1}}$ accepts)
$\left[\begin{array}{l} (< \overline{n}, w >, w') \\ \mapsto (< \overline{n+1}, w >, w') \\ \textbf{where} \ \ w, w' \in \Sigma^*, n \in \mathbb{N} \end{array} \right]$
end while
Accept

P_k can be ω- clocked: mark all the instructions but those that were not marked in $\tilde{P_{k-1}}$. Take as time period the maximum of 1 and of the time period of $\tilde{P_{k-1}}$. P_k is of dimension $d + k' - 1$.

□

Lemma 3.7 *Assume that B is a discrete language such that all the languages of Σ_1^B are semi-recognized by an ω-clocked RCT program in dimension $d' \geq 2$.*

Let S be a discrete language with $S \in \Delta_k^B$, $k \in \mathbb{N}, k \geq 2$. Then S is fully-recognized by a RCT program in dimension $d' + k - 1$.

Proof: By definition of Δ_k^B, there must exists $S', S'' \in \Sigma_{k-1}^B$ such that $x \in S \Leftrightarrow \exists n \in \mathbb{N} \ \ < \overline{n}, x > \notin S'$ and $x \notin S \Leftrightarrow \exists n \in \mathbb{N} \ \ < \overline{n}, x > \notin S''$: see [16].

By lemma 3.6, S and S' are respectively semi-recognized by some ω-clocked programs P_{k-1}, P'_{k-1} in dimension $k + d' - 2$. Denote $\tilde{P_{k-1}}$, $\tilde{P'_{k-1}}$ the programs that one gets by applying lemma 3.5 on P_{k-1} and P'_{k-1} respectively. S is fully recognized by the following RCT program M_k:

Algorithm 10 Program M_k

$\left[\begin{array}{l} (w, w') \\ \mapsto (< \overline{0}, w >, w') \\ \textbf{where} \ \ w, w' \in \Sigma^* \end{array} \right]$
while ($\tilde{P_{k-1}}$ accepts) **do**
  if ($\tilde{P'_{k-1}}$ accepts) **then**
    $\left[\begin{array}{l} (< \overline{n}, w >, w') \\ \mapsto (< \overline{n+1}, w >, w') \\ \textbf{where} \ \ w, w' \in \Sigma^*, n \in \mathbb{N} \end{array} \right]$
  else
    Reject
  end if

end while
Accept

□

Theorem 3.4 *Let S be a discrete language.*

- *Assume $S \in \Sigma_k$, $k \geq 1$. Then S is semi-recognized by an ω-clocked RCT program of dimension $1+k$.*
- *Assume $S \in \Delta_k$, $k \geq 1$. Then S is fully-recognized by a RCT program of dimension $1+k$.*

Proof: Immediate from theorem 3.1, and from lemma 3.6 and lemma 3.7 respectively with $B = \emptyset$.

□

3.4 RCT machines and the hyper-arithmetical hierarchy

3.4.1 Realizing any 2PDA program in time kx_{d+1}

We prove first a technical lemma: if a 2PDA can do some job, one can build a RCT machine of dimension $d+1 \geq 3$ that does the same job but in time bounded by kx_{d+1} for some k:

Lemma 3.8 *Let $d \geq 2$. Let M be an ω-2PDA. Assume that, for all $w_1 \in \Sigma^\omega, w_2 \in \Sigma^*$, M maps (w_1, w_2) to $(f_1(w_1, w_2), f_2(w_1, w_2)) \in \Sigma^\omega \times \Sigma^*$.*

There exists some $k_M \in \mathbb{R}^+$ and a RCT machine M' of dimension $d+1$ that, for all $w_1 \in \Sigma^\omega, w_2 \in \Sigma^*$, for all $y_3, \ldots, y_d \in [0,1]$, for all $n \in \mathbb{N}$, maps

$$(\mathcal{J}(w_1)/2^n, \mathcal{J}(w_2)/2^n, y_3/2^n, \ldots, y_d/2^n, 1/2^n)$$

to

$$(\mathcal{J}(f_1(w_1, w_2))/2^n, \mathcal{J}(f_2(w_1, w_2))/2^n, y_3/2^n, \ldots, y_d/2^n, 1/2^n)$$

in a time bounded above by $k_M/2^n$.

Proof: The idea is to build a machine that simulates M, that does some $Div2^{d+1}$ instructions every two steps and that counts in parallel the number of $Div2^{d+1}$ instructions already executed. The machine simulates M until M accepts. At this moment, the machine does some $Mul2^{d+1}$ instructions in order to come back to $x_{d+1} = 1/2^n$.

Let $\uparrow$ be a letter of Σ. If $i \in \{1,2\}$, denote $\tilde{i}$ for $3-i$. Assume without loss of generality that any ID of any computation of M on any input is of type (q, w_1, w_2), $q \in Q, w_1, w_2 \in \Sigma^*$ with the first letter of w_1 and the first letter of w_2 different from letter $\uparrow$.

Replace one after the other the 2PDA instructions of M by some new instructions using the following correspondence, , where RCT instructions $Top^i(j)?$, $Pop^i(j)$, $Push^i(j)$ are defined in figure 4:

Old ω-2PDA instruction	New RCT instructions
$Pop^1(j), j \in \Sigma$	$Transfer_2; (Pop^1(j))/x_{d+1}; (Push^1(\uparrow))/x_{d+1};$ $Div2^{d+1}; Transfer_1$
$Push^1(j), j \in \Sigma$	$Transfer_2; (Push^1(j))/x_{d+1};$ $(Push^1(\uparrow))/x_{d+1}; Div2^{d+1}; Transfer_1$
$Top^1(j)?, j \in \Sigma$	$Transfer_2; (Top^1(j)?)/x_{d+1};$ $(Push^1(\uparrow))/x_{d+1}; Div2^{d+1}; Transfer_2$
$Pop^2(j), j \in \Sigma$	$(Pop^2(j))/x_{d+1}; (Push^1(\uparrow))/x_{d+1};$ $Div2^{d+1}$
$Push^2(j), j \in \Sigma$	$(Push^2(j))/x_{d+1}; (Push^1(\uparrow))/x_{d+1}; Div2^{d+1}$
$Top^2(j)?, j \in \Sigma$	$(Top^2(j)?)/x_{d+1}; (Push^1(\uparrow))/x_{d+1}; Div2^{d+1}$

where $Transfer_i$, $i \in \{1,2\}$ transfers all the $\uparrow$ from register $\tilde{i}$ to register i and is the following sequence of instructions:

Algorithm 11 $Transfer_i$

while $((Top^{\tilde{i}}(\uparrow)?)/x_{d+1})$ **do**
  $(Pop^{\tilde{i}}(\uparrow))/x_{d+1}$
  $(Push^{i}(\uparrow))/x_{d+1}$ /* *Transfer one* $\uparrow$ *from stack* $\tilde{i}$ *to stack* i*/
  $(Push^{i}(\uparrow))/x_{d+1}$
  $Div2^{d+1}$
  $(Push^{i}(\uparrow))/x_{d+1}$
  $Div2^{d+1}$
  $(Push^{i}(\uparrow))/x_{d+1}$
  $Div2^{d+1}$ /* *Add 3* $\uparrow$ *since the transfer of the* $\uparrow$ *was done with 3 instructions*/
end while

One gets a RCT program P_M of dimension $d+1$: this program simulates M, does some $Div2^{d+1}$ instructions every two steps and has the following property: at every step of the simulation of M, the value of the first real register of P_M is of type $\mathcal{J}(\uparrow^p\ w)$, $w \in \Sigma^*$, where $p \in \mathbb{N}$, is the number of instructions $Div2^{d+1}$ already executed by P_M.

Consider now M' as the following RCT program:

Algorithm 12 Program M'

P_M /* *Map* $x_1 = \mathcal{J}(w_1)/2^n$, $x_2 = \mathcal{J}(w_2)/2^n$, $x_3 = y_3/2^n$, ..., $x_d = y_d/2^n$, $x_{d+1} = 1/2^n$ *to* $x_1 = \mathcal{J}(\uparrow^p\ f_1(w_1,w_2))/2^{n+p}$, $x_2 = \mathcal{I}(f_2(w_1,w_2))/2^{n+p}$, $x_3 = y_3/2^{n+p}$,..., $x_d = y_d/2^{n+p}$, $x_{d+1} = 1/2^{n+p}$).*/

while $((Top^1(\uparrow)?)/x_{d+1})$ **do**
  $(Pop^1(\uparrow))/x_{d+1}$ /* *Get back to* $x_{d+1} = 1/n$ *by undoing the* $Div2^{d+1}$ *instructions*/
  $Mul2^{d+1}$
end while

Program M' is executed in a time bounded above by $k_M 1/2^n$ for some fixed $k_M \in \mathbb{R}^+$.

□

Convention 3.2 *We denote by*

$$\left(\begin{array}{l} (w_1, w_2) \\ \mapsto (w_1', w_2') \\ \textbf{where} \quad \text{“conditions”} \end{array} \right) | x_{d+1}$$

any RCT program of dimension $d+1$ *given by lemma 3.8 that for all* $w_1, w_1' \in \Sigma^\omega$, $w_2, w_2' \in \Sigma^*$ *verifying “conditions”, and for all* $y_3, \ldots, y_d \in [0,1]$, *for all* $n \in \mathbb{N}$, *maps* $(\mathcal{J}(w_1)/2^n, \mathcal{J}(w_2)/2^n, y_3/2^n, \ldots, y_d/2^n, 1/2^n)$ *to* $(\mathcal{J}(w_1')/2^n, \mathcal{J}(w_2')/2^n, y_3/2^n, \ldots, y_d/2^n, 1/2^n)$ *in a time bounded by* $k/2^n$ *for some* k.

3.4.2 Setting the m^{th} digit of a real in time $k/2^m$ for some k

The following lemma is the main trick that will be used in lemma 3.10 to show that one can recognize some hyper-arithmetical sets: one can build a RCT machine of dimension $d+2$ that, on input $m \in \mathbb{N}$, add $1/2^m$ to real register x_{d+2} in a time proportional to $maximum(1/2^m, x_{d+1})$:

Lemma 3.9 *Let $\#, \$ \in \Sigma$ be two distinct letters of Σ used as delimiters.*

For all $d \geq 2$, there exists some $k \in \mathbb{R}^+$ and a RCT machine $WriteDigit_{d+2}$ of dimension $d+2$ that, for all $y_3, \ldots, y_d, y_{d+2} \in [0,1], m, n \in \mathbb{N}, w \in \Sigma^\omega, w' \in \Sigma^*$, maps

$$(\mathcal{J}(\#^n\overline{m}\$w)/2^n, \mathcal{I}(w')/2^n, y_3/2^n \ldots, y_d/2^n, 1/2^n, y_{d+2})$$

to

$$(\mathcal{J}(\#^n\overline{m}\$w)/2^n, \mathcal{I}(w')/2^n, y_3/2^n, \ldots, y_d/2^n, 1/2^n, y_{d+2} + 1/2^m)$$

in a time upper bounded by $k1/2^{minimum(m,n)}$.

Proof: The general idea is to do some $Mul2^{d+1}/Div2^{d+1}$ instructions in order to get $x_{d+1} = 1/2^m$, then to do a $x_{d+2} := x_{d+2} + x_{d+1}\ [x_{d+1}]$ instruction, and then to do some $Div2^{d+1}/Mul2^{d+1}$ instructions to come back to $x_{d+1} = 1/2^n$.

Assume without loss of generality that one can find two distinct letters $\uparrow$ and $\downarrow$ in Σ different from letter \$ and from letter #.

$WriteDigit_{d+2}$ is the following RCT program , where RCT instructions $Top^i(j)?$ $Pop^i(j)$, $Push^i(j)$ are defined in figure 4:

Algorithm 13 $WriteDigit_{d+2}$

$$\left(\begin{array}{l} (\#^n\overline{m}\$w, w') \\ \mapsto (move_1\$move_2\$w'\$\#^n\overline{m}\$w, \epsilon) \\ \qquad w \in \Sigma^\omega, w' \in \Sigma^*, m, n \in \mathbb{N} \\ \qquad move_1, move_2 \in \Sigma^* \\ \textbf{where} \quad (move_1, move_2) = \left\{\begin{array}{ll} (\downarrow^{m-n}, \uparrow^{m-n}) & \text{if } m > n \\ (\uparrow^{n-m}, \downarrow^{n-m}) & \text{if } m < n \\ (\epsilon, \epsilon) & \text{if } m = n \end{array}\right. \end{array}\right) |x_{d+1}$$

$$/*\ Map \left\{\begin{array}{l} x_1 = \mathcal{J}(\#^n\overline{m}\$w)/2^n \\ x_2 = \mathcal{I}(w')/2^n \\ x_{d+1} = 1/2^n \end{array}\right. to$$

$$\left\{\begin{array}{l} x_1 = \mathcal{J}(move_1\$move_2\$w'\$\#^n\overline{m}\$w) \\ \quad /2^n \\ x_{d+1} = 1/2^n \end{array}\right.$$

in a time bounded by $k_1 1/2^n$ for some k_1: see lemma 3.3./*

GoUpOrDown — */*Call some $Mul2^{d+1}/Div2^{d+1}$ instructions to get $x_{d+1} = 1/2^m$*/*

$x_{d+2} := x_{d+2} + x_{d+1}\ [x_{d+1}]$ — */* Add $1/2^m$ to x_{d+2}*/*

GoUpOrDown — */*Call some $Div2^{d+1}/Mul2^{d+1}$ instructions to get $x_{d+1} = 1/2^n$*/*

$$\left(\begin{array}{l} (w'\$\#^n\overline{m}\$w, \epsilon) \\ \mapsto (\#^n\overline{m}\$w, w') \\ \textbf{where} \quad w \in \Sigma^\omega, w' \in \Sigma^*, m, n \in \mathbb{N} \end{array}\right) |x_{d+1}$$

*/*Set $x_1 = \mathcal{J}(\#^n\overline{m}\$w)/2^n$, $x_2 = \mathcal{I}(w')/2^n$ in time bounded by $k_2 1/2^n$ for some k_2. */*

---

where program *GoUpOrDown* is the following RCT program:

Algorithm 14 program *GoUpOrDown*

---

if $(Top^1(\uparrow))/x_{d+1}$
then
while $((Top^1(\uparrow))/x_{d+1})$ **do**
$(Pop^1(\uparrow))/x_{d+1}$
$Mul2^{d+1}$
end while
end if
if $(Top^1(\downarrow))/x_{d+1}$
then
while $((Top^1(\downarrow))/x_{d+1})$ **do**
$(Pop^1(\downarrow))/x_{d+1}$
$Div2^{d+1}$
end while
end if
$(Pop^1(\$))/x_{d+1}$

The execution of the calls to program $GoUpOrDown$ are done in a time upper bounded by $k_3 1/2^{min(m,n)}$ for some k_3. As a consequence, there exists some $k \in \mathbb{R}^+$ such that the whole execution of program $WriteDigit_{d+2}$ is done in a time bounded above by $k1/2^{minimum(m,n)}$.

□

3.4.3 Outputting reals encoding languages

In definition 2.3, we defined mapping $\mathcal{J}$ that encodes any word of Σ^ω into a real of $[0,1]$. Now, we define mapping $\mathcal{L}$ that encodes any discrete language $L \subset \Sigma^*$ into a real of $[0,1]$:

Definition 3.10 (Encoding by $\mathcal{L}$) *Let $\Sigma = \{0, 1, \ldots, n_\Sigma\}$ be the fixed finite alphabet. Let α and β be two letters of Σ with $\alpha \neq \beta$.*

- *Let $L \subset \Sigma^*$ be a discrete language. We denote by w_L the infinite word $a_0 a_1 a_2 \ldots a_i \ldots$ such that, for all $i \in \mathbb{N}$, $a_i = \alpha$ (respectively: $a_i = \beta$) iff the i^{th} word of Σ^* is in L (resp. is not in L)*
- *Denote by $\mathcal{P}(\Sigma^*)$ the set of the subsets of Σ^*.*

  We denote by $\mathcal{L}$ the mapping from $\mathcal{P}(\Sigma^*)$ to $[0,1]$ that, for all $L \subset \Sigma^*$, maps L to $\mathcal{L}(L) = \mathcal{J}(w_L)$.

Check that $\mathcal{J}(w_\emptyset)$ is in $\mathbb{Q}$ and that a machine can do with any problem $x_1 := \mathcal{L}(\emptyset)$.

Using lemma 3.9, we show that if we can enumerate a set then we can output a real encoding this set in finite time:

Lemma 3.10 *Let $d \geq 2$. Let $\$, \#$ be two letters of Σ used as delimiters.*

Assume we have a function $f : \mathbb{N} \times \Sigma^* \times \Sigma^\omega \to \Sigma^*$, a constant $k_f \in \mathbb{R}^+$ and a RCT machine M_f of dimension $d+1$ that, for all $n \in \mathbb{N}, w \in \Sigma^*, L \subset \Sigma^*, y_2 \in I$, maps

$$(\mathcal{J}(\#^n \$ w \$ w_L)/2^n, y_2/2^n, ., \ldots, ., 1/2^n)$$

to

$$(\mathcal{J}(f(n, w, w_L) \$ w_L)/2^n, y_2/2^n, ., \ldots, ., 1/2^n)$$

in a time bounded above by $k_f 1/2^n$.

Then there exists a RCT machine M'_f of dimension $d+2$ that for all discrete language $L \subset \Sigma^*$, for all word $w \in \Sigma^*$ and real $y_2 \in I$, maps $(\mathcal{J}(w\$w_L), y_2, \ldots.)$ to $(\mathcal{J}(w\$w_{L(\tilde{w})}), y_2, 0, \ldots, 0)$ in a bounded time, where $L(\tilde{w}) = \{w' | w' \in \Sigma^* \ \wedge \ \exists n \in \mathbb{N}\ f(n, w, w_L) = w'\}$

Proof: The general idea is to write a program that, on input $x_1 = \mathcal{J}(w\$w_L)$, $x_2 = \mathcal{I}(w')$, using lemma 3.9, writes digit by digit onto its real register x_{d+2} the real value of $\mathcal{J}(w\$w'\$w_{\tilde{L(w)}})$.

Denote by $number : \Sigma^* \to \mathbb{N}$ the function that maps any word $w \in \Sigma^*$ onto its number in the enumeration of the words of Σ^*. For $k \in \mathbb{N}, w \in \Sigma^*$, denote $lttr(w,k)$ for k^{th} letter of word w. We assume fixed a recursive enumeration of the finite subsets of Σ^* similar to the one of [16]: for any integer $n \in \mathbb{N}$, D_n denotes the n^{th} finite subset of Σ^*.

M'_f is given by the following algorithm, where RCT instructions $Top^i(j)$, $Pop^i(j)$,$Push^i(j)$ are defined in figure 4 and integer b'_Σ is defined page 3:

Algorithm 15 Program M'_f

$$\left[\begin{array}{ll} & (w\$w_L, w') \\ & \mapsto (\$w_L, w\$w'\$\overline{u_0}) \\ \textbf{where} & w_L \in \Sigma^\omega, w, w' \in \Sigma^* \\ & u_0 \in \mathbb{N}, D_{u_0} = \emptyset \end{array}\right]$$

/*Initialize the computation: set $n = 0$*/

$x_{d+1} := 1$ /*Set initial speed to 1*/

$x_{d+2} := 0$ /*Set x_{d+2} to 0*/

while (true) do /*While (true)*/

$$\left(\begin{array}{ll} & (\#^n\$w_L, w\$w'\$\overline{u_n}) \\ & \mapsto (\#^n\overline{b'_\Sigma * n - 2a_n}\$w_L, w\$w'\$\overline{u_n}) \\ & w_L \in \Sigma^\omega, w, w' \in \Sigma^*, \\ & n, u_n \in \mathbb{N}, a_n \in \Sigma \\ \textbf{where} & a_n = \left\{\begin{array}{ll} \alpha & \text{if } n > length(w\$w'\$) \\ \text{lttr(w\$w'\$,n)} & \text{if } n \leq length(w\$w'\$) \end{array}\right. \end{array}\right) |x_{d+1}$$

$WriteDigit_{d+2}$ /*Set the n^{th} digit of x_{d+2} to default value a_n*/

$$\left(\begin{array}{l} (\#^n\overline{m}\$w_L, w\$w'\$\overline{u_n}) \\ \mapsto (\#^n\$w\$w_L, \#^n\$w\$w'\$\overline{u_n}) \\ \textbf{where}\ \ w_L \in \Sigma^\omega, w, w' \in \Sigma^*, n, u_n \in \mathbb{N} \end{array}\right) |x_{d+1}$$

M_f /*Get w'' the n^{th} word of the enumeration given by M_f*/

$$\left(\begin{array}{ll} & (w''\$w_L, \#^n\$w\$w'\$\overline{u_n}) \\ & \mapsto (\#^n\overline{m}\$w_L, already in\$w\$w'\$\overline{u_{n+1}}) \\ & w, w', w'' \in \Sigma^*, w_L \in \Sigma^\omega, \\ & n, u_n, u_{n+1} \in \mathbb{N}, already in \in \Sigma \\ & m = b'_\Sigma * (number(w) + length(w\$w'\$) + 1) \\ \textbf{where} & \ -2(\beta - \alpha) \\ & \text{if } w'' \in D_{u_n} \text{ then} \\ & \ already in = \#,\ u_{n+1} = u_n \\ & \text{else} \\ & \ already in = \$, D_{u_{n+1}} = D_{u_n} \cup \{w''\} \end{array}\right) |x_{d+1}$$

if $((Top^2(\$))/x_{d+1})$ **then** /*If word w'' has not been yet output*/

$(Pop^2(\$))/x_{d+1}$

$WriteDigit_{d+2}$ /*Then change the digit of real register x_{d+2} corresponding to w'' from value α to value β. */

else

$(Pop^2(\#))/x_{d+1}$ /*Else do nothing*/

end if

$$\left(\begin{array}{l}(\#^n\overline{m}\$w_L, \$w\$w'\$\overline{u_{n+1}})\\ \mapsto (\#^{n+1}\$w_L, w\$w'\$\overline{u_{n+1}})\\ \textbf{where}\ \ w_L \in \Sigma^\omega, w, w' \in \Sigma^*, u_{n+1}, n, m \in \mathbb{N}\end{array}\right) | x_{d+1}$$

/*Do $n := n + 1$*/

$Div2^{d+1}$

end while

$limit^*$:

$x_1 := x_{d+2}$ /*Copy the result into x_1*/

$x_{d+2} := 0$ /*Set x_{d+2} to 0*/

$$\left[\begin{array}{l}(w\$w'\$w_L, \epsilon)\\ \mapsto (w\$w_L, w')\\ \textbf{where}\ \ w_L \in \Sigma^\omega, w, w' \in \Sigma^*\end{array}\right]$$

/*Put back the result in the good form*/

□

3.4.4 Climbing up the hyper–arithmetical hierarchy

We start by an easy lemma used in lemma 3.12:

Lemma 3.11

- *For all $y \in O$, $\{\overline{x} | x \in O \ \wedge \ x <_0 y\}$ is recursively enumerable uniformly in y: there exists a recursive $f : \mathbb{N} \to \mathbb{N}$ such that for all $y \in O$, the range of $\phi_{f(y)}$ is $\{\overline{x} | x <_0 y\}$*
- *Given $x, y \in O$ with $x \leq_o y$ or $y \leq_0$ as input, a Turing machine can effectively tell if $x = y$, if $x <_0 y$ or if $y <_0 x$.*
- *There exists a recursive l such that, for all z_1, z_2 with $z_1 \leq_0 z_2$, $H(z_1) = W^{H(z_2)}_{l(z_1,z_2)}$*

Proof: See [16] for the first assertion. For the second assertion enumerate in parallel the predecessors of x and of y until x or y is found. For all z_1, z_2, set $l(z_1, z_2)$ as number of the Turing machine with oracle that, on any input w, test the membership of word $\phi_{h(z_1,z_2)}(w)$ to its oracle, where h is the recursive function of lemma 2.2.

□

Now, we apply recurrently lemma 3.10 in order to get some machines that output $\mathcal{L}(L)$ for some discrete languages $L \in \Sigma^*$ in higher and higher levels of the hyper-arithmetical hierarchy. We define ω^0 as 1.

Lemma 3.12 *Let $k \geq 0$.*

There exists $z_k \in O$ with $|z_k| = \omega^k$, there exists $f_k : \mathbb{N} \times \Sigma^* \times \Sigma^\omega \to \Sigma^*$, there exists some fixed $C_k \in \mathbb{R}^+$ and:

- *a RCT machine M_k of dimension $2k + 2$ that, for all $n \in \mathbb{N}, w \in \Sigma^*, L \subset \Sigma^*, y_2 \in I$, maps $(\mathcal{J}(\#^n\$w\$w_L), y_2, ., \ldots, .)$ to $(\mathcal{J}(f_k(n, w, w_L)\$w_L), y_2, ., \ldots, .)$*
- *a RCT machine M'_k of dimension $2k + 3$ that, for all $n \in \mathbb{N}, w \in \Sigma^*, L \subset \Sigma^*, y_2 \in I$ maps $(\mathcal{J}(\#^n\$w\$w_L)/2^n, y_2/2^n, ., \ldots, ., 1/2^n)$ to $(\mathcal{J}(f_k(n, w, w_L)\$w_L), y_2/2^n, ., \ldots, ., 1/2^n)$ in a time bounded above by $C_k/2^n$.*

such that, for all $z \in O$, if $L = H(z)$ then $L(\tilde{\overline{z}}) = \{w' | w' \in \Sigma^* \ \wedge \ \exists n \in \mathbb{N} \ f(n, \overline{z}, w_L) = w'\} = H(z +_0 z_k)$

Proof: It is known that there exists a recursive g such that for all $L \in \Sigma^*, m \in \mathbb{N}$, the range of function $\phi^L_{g(m)}$ is is W^L_m [16]. Let M_{univ} be an ω-2PDA such that on input $\#^n\$\overline{m}$, M_{univ} simulates $M^L_{g(m)}$ on input n, answering the queries of $M^L_{g(m)}$ on any word w' to its oracle L by comparing the digit of w_L corresponding to w' to letter α. M_{univ} is an ω-2PDA that for all $n \in \mathbb{N}, m \in \mathbb{N}, w' \in \Sigma^*, L \subset \Sigma^*$, maps $(\#^n\$\overline{m}\$w_L, w')$ to $(w^m_n\$w_L, w')$, where $w^m_n = \phi^L_{g(m)}(n)$.

Denote by P_{univ} the RCT machine given by lemma 3.1 that simulates M_{univ}.

Using lemma 3.8, for all $d \geq 2$ one can build a RCT machine P'^{d+1}_{univ} of dimension $d+1$ that, for all $n \in \mathbb{N}, m \in \mathbb{N}, y_2 \in I, L \subset \Sigma^*$, maps $(\mathcal{J}(\#^n\$\overline{m}\$w_L)/2^n, y_2/2^n, ., \ldots, ., 1/2^n)$ to $(\mathcal{J}(w_m^n\$w_L)\ /2^n, y_2/2^n, ., \ldots, ., 1/2^n)$ in time $k/2^n$ for some fixed $k \in \mathbb{R}^+$. Apply lemma 3.10 on this machine: one gets a RCT machine of dimension $d+2$ that, for all $L \subset \Sigma^*, m \in \mathbb{N}, y_2 \in I$, maps $(\mathcal{J}(\overline{m}\$w_L), y_2, ., \ldots, .)$ to $(\mathcal{J}(\overline{m}\$w_{W_m^L}), y_2, 0, \ldots, 0)$ in finite time. Denote this RCT machine by P''^{d+2}_{univ}.

Now, we are ready to prove the assertions of the lemma by induction over k:

Assume $k = 0$: it is known that there exists $m_0 \in \mathbb{N}$, such that for all $L \subset \Sigma^*$, $L' = W^L_{m_0}$ [16]. Consider M as the ω-2PDA that on input $(\#^n\$w\$w_L, w')$ calls M_{univ} with input $(\#^n\$\overline{m_0}\$w_L, w')$. M_0 is the RCT machine of dimension 2 given by lemma 3.1 that simulates M, and M'_0 is the RCT machine of dimension 3 given by lemma 3.8 that simulates M.

Assume now $k \geq 1$: denote by $\Pi_1, \Pi_2, \Pi_3 : \mathbb{N} \to \mathbb{N}$ some recursive functions such that

$$n \mapsto (\Pi_1(n), \Pi_2(n), \Pi_3(n))$$

is a bijective recursive function from $\mathbb{N}$ to $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$. Denote by f and l the recursive functions of lemma 3.11.

Lemma 3.10 can be applied on machine M'_{k-1}: one get a RCT machine M''_{k-1} of dimension $2k+2$ that for all $z' \in O$, for all $y_2 \in I$ maps $(\mathcal{J}(\overline{z'}\$w_{H(z')}), y_2, ., \ldots, .)$ to $\mathcal{J}(\overline{z'}\$w_{H(z'+z_{k-1})}, y_2, 0, \ldots, 0)$ in a bounded time. Set $z_k = 3.5^{n_k}$ where $\phi_{n_k}(0) = 1$ and $\phi_{n_k}(n+1) = \phi_{n_k}(n) +_0 z_{k-1}$ for all $n \in \mathbb{N}$.

M_k is given by the following program, where RCT instructions $Top^i(j)$, $Pop^i(j)$,$Push^i(j)$ are defined in figure 4:

Algorithm 16 Program M_k

$$\left[\begin{array}{l} (\#^n\$\overline{z}\$w_L, w') \\ \mapsto (\overline{z}\$w_L, \#\$\#^n\$\overline{z}\$w') \\ \textbf{where}\ \ w_L \in \Sigma^\omega, w' \in \Sigma^*, z \in O, n \in \mathbb{N} \end{array}\right]$$

while $(Top^2(\#)?)$ **do** /* *Call* $\Pi_1(n)$ *times program* M''_{k-1} */

M''_{k-1}

$$\left[\begin{array}{ll} (\overline{z_p}\$w_L, \#\$\#^n\$\overline{z}\$w') & \\ \mapsto (\overline{z_{p+1}}\$w_L, continue\$\#^n\$\overline{z}\$w') & \\ & w_L \in \Sigma^\omega, w' \in \Sigma^*, z_p, z_{p+1} \in O, n \in \mathbb{N} \\ \textbf{where} & z_{p+1} = z_p +_0 y_{k-1} \\ & \text{if } z_{p+1} \leq_o \phi_{n_k}(\Pi_1(n)) \\ & \text{ then } continue = \# \\ & \text{ else } continue = \$ \end{array}\right]$$

end while /**Here, if the initial input was* $\#^n\$\overline{z}\$w_{H(z)}$, $z \in O$, *we have* $x_1 = \mathcal{J}(\overline{z_p}\$w_{H(z_p)})$ *where* $z_p = \phi_{n_k}(\Pi_1(n))$ */

$$\left[\begin{array}{ll} (\overline{z_p}\$w_L, \$\$\#^n\$\overline{z}\$w') & \\ \mapsto (\#^{\Pi_2(n)}\$\overline{m}\$w_L, \#^n\$\overline{z}\$w'\$\overline{z''}\$\overline{z_p}) & \\ & w_L \in \Sigma^\omega, w' \in \Sigma^*, z \in O, m, n \in \mathbb{N} \\ \textbf{where} & z'' = \phi_{f(z_p)}(\Pi_3(n)) \text{ (we have } z' \leq_0 z_p) \\ & m = l(z'', z_p) \text{ is the integer such that } W_m^{H(z_p)} = H(z'') \end{array}\right]$$

P_{univ} /* *Compute* $z'' = \phi_{f(z_p)}(\Pi_3(n))$. *We have* $z'' \leq_o z_p$. *Get* w'' *the* $\Pi_2(n)^{th}$ *word of* $H(z'')$*/

$$\left[\begin{array}{l}(w''\$w_L, \#^n\$\overline{z}\$w'\$\overline{z''}\$\overline{z_p}) \\ \mapsto (\overline{m}\$w_L, \#^n\$w''\$\overline{z''}\$w') \\ \textbf{where} \quad w_L \in \Sigma^\omega, w', w'' \in \Sigma^*, z, z'', z_p \in O, m, n \in \mathbb{N} \\ \qquad\quad m = l(z, z_p) \text{ is the integer such that } W_m^{H(z_p)} = H(z)\end{array}\right]$$

$P_{univ}^{''2k+2}$ /*Put back in x_1 the value of $w_{H(z)}$*/

$$\left[\begin{array}{l}(\overline{m}\$w_L, \#^n\$w''\$\overline{z''}\$w') \\ \mapsto (< w'', z'' > \$w_L, w') \\ \textbf{where} \quad w_L \in \Sigma^\omega, w', w'' \in \Sigma^*, z'' \in O, n \in \mathbb{N}\end{array}\right]$$

/*Output $< w'', z'' > \$w_{H(z)}$ */

M'_k is easy to obtained from the program of M_k: add the instruction $x_{2k+3} := 1[1]$ at the beginning of program M_k, replace

$$\left[\begin{array}{l}(\overline{m}\$w_L, \#^n\$w''\$\overline{z''}\$w') \\ \mapsto (< w'', z'' > \$w_L, w') \\ \textbf{where} \quad w_L \in \Sigma^\omega, w', w'' \in \Sigma^*, z'' \in O, n \in \mathbb{N}\end{array}\right]$$

by

$$\left(\begin{array}{l}(\overline{m}\$w_L, \#^n\$w''\$\overline{z''}\$w') \\ \mapsto (< w'', z'' > \$w_L, \uparrow^{\Pi_1(n)} \$w') \\ \textbf{where} \quad w_L \in \Sigma^\omega, w', w'' \in \Sigma^*, z'' \in O, n \in \mathbb{N}\end{array}\right) | x_{d+1}$$

replace in program M_k all the other instructions of type

$$\left[\begin{array}{l}(w_1, w_2) \\ \mapsto (w'_1, w'_2) \\ \textbf{where} \quad \text{conditions}\end{array}\right]$$

by

$$\left(\begin{array}{l}(w_1, w_2) \\ \mapsto (w'_1, w'_2) \\ \textbf{where} \quad \text{conditions}\end{array}\right) | x_{d+1}$$

replace P_{univ} by $P_{univ}^{'2k+3}$, $P_{univ}^{''2k+2}$ by $P_{univ}^{''2k+2}/x_{2k+3}$, and replace the call to M''_{k-1} by the instructions $M''_{k-1}; Div 2^{2k+3}$, and add the program $GoUpOrDown$ defined page 20 at the end of the program.

□

We get:

Lemma 3.13 *Let $k \geq 1$.*

- *Any language of Δ_{ω^k} can be fully-recognized by a RCT machine of dimension $2k+2$.*
- *Any language of Σ_{ω^k} can be semi-recognized by a RCT machine of dimension $2k+2$.*

Proof: Consider the machine M_k and the integer $z_k \in O$ of lemma 3.12. M_k is of dimension $2k+2$, and $|z_k| = \omega^k$. Let L be a language of Σ_{ω^k} (respectively: Δ_{ω^k}). L is recursively enumerable (resp. recursive) in $H(z_k)$ by some machine $M_n^{H(z_k)}$.

See that there exists a recursive g such that, for all $u \in \Sigma^*, v \in O$, $u \notin H(v) \Leftrightarrow g(u) \in H(2^v)$: see [16].

L is semi-recognized (resp. fully-recognized) by the RCT machine of dimension $2k+2$ that simulates $M_n^{H(z_k)}$, simulating every query of $M_n^{H(z_k)}$ of type $< u, v > \in H(z_k)$? by a subprogram that runs M_k on input $x_1 = \mathcal{J}(\#^n\$\overline{1}\$w_\emptyset)$ for $n = 1, 2, \ldots$, until either $x_1 = \mathcal{J}(< u, v > \$w_\emptyset)$ or $x_1 = \mathcal{J}(< g(u), 2^v > \$w_\emptyset)$ is output.

□

We are ready to prove the main assertion of the section: RCT machine can recognize some hyper-arithmetical sets. We define $\omega^0 = 1$.

Theorem 3.5 *Let $k \geq 0$.*

- *Any language of Δ_{ω^k} can be fully-recognized by a RCT machine of dimension $2k+2$.*
- *Any language of Δ_{ω^k+1} can be fully-recognized by a RCT machine of dimension $2k+3$.*
- *Any language of Σ_{ω^k} can be semi-recognized by a RCT machine of dimension $2k+2$.*
- *Any language of Σ_{ω^k+1} can be semi-recognized by a RCT machine of dimension $2k+3$.*

Proof: If $k = 0$, this is a direct application of theorem 3.4.

Assume now $k \geq 1$: the first and the third assertions are lemma 3.13. The second and the last assertions are immediate from the third assertion and from lemmas 3.7 and 3.6 with $B = H(\omega^k)$, since $\Sigma_{\omega^k+1} = \Sigma_2^{H(\omega^k)}$ and $\Delta_{\omega^k+1} = \Delta_2^{H(\omega^k)}$.

□

4 PCD systems can simulate RCT Machines

In this section, we prove that PCD systems can simulate RCT machines. We start by seeing how to realize the elementary instructions of RCT machines.

4.1 Linear machine instructions

Let d be an integer. A *k-dimensional box of $\mathbb{R}^d$*, $k < d$, is a couple $I = (P, B)$ where P is a polyhedral subset of $\mathbb{R}^d$ of dimension k, and B is a affine basis $(O, e_1, e_2, \ldots, e_d)$ of $\mathbb{R}^d$, $O \in P$, such that $(O, e_1, \ldots, e_k)$ is an affine basis[1] of P.

The *point of coordinates $(x_1, \ldots, x_k)$ on $I = (P, B)$* denotes the point of P, if it exists, of coordinates $(x_1, \ldots, x_k, 0, \ldots, 0)$ in basis B. A trajectory is said to reach $I = (P, B)$ iff it reaches P.

Let $\mathcal{H}$ be a PCD system of dimension d. Let d' be an integer with $d' < d$. Let $I = (f, c)$ be an assignment[2] of dimension d'. $\mathcal{H}$ is said to *realize assignment I* if there exist some d'-dimensional boxes In and Out of $\mathbb{R}^d$, such that, for all $x \in [0,1]^{d'}$, the trajectory of $\mathcal{H}$ starting from the point of coordinates $x \in [0,1]^{d'}$ on In at time 0 reaches Out at time $c(x)$ in point of coordinates $f(x)$ on Out: see figure 5 and figure 6. In that case, we say that $\mathcal{H}$ realizes the assignment *via input port In and via output port Out*.

For all $d' \in \mathbb{N}$, denote by $Id_{d'}$ the identity function of $[0,1]^{d'}$: $Id_{d'}(x) = x$ for all $x \in [0,1]^{d'}$. Let $I = (R, c)$ be a test of dimension d'. $\mathcal{H}$ is said to *realize test I* if there exist three d'-dimensional boxes In, Out^+,Out^- of $\mathbb{R}^d$ such that for all x such that $R(x)$ is true, $\mathcal{H}$ realizes assignment $(Id_{d'}, c)$ via input port In and output port Out^+, and for all x such that $R(x)$ is false, $\mathcal{H}$ realizes assignment $(Id_{d'}, c)$ via input port In and output port Out^-.

Lemma 4.1 (Basic linear machine instructions) *Let $d \in \mathbb{N}$. Let $d' \geq d+1$.*

Let $I = (f, c) \in Assgnmt_d$ be an admissible assignment (respectively Let $I = (R, c) \in Test_d$ be an admissible test) of dimension d. Assume that I is one of the "linear machine instructions" of definition 3.5.

For all $\mu \in \mathbb{R}^+$, one can build a PCD system of dimension d' that realizes assignment $I = (f, \mu c)$ (resp. that realizes test $I = (R, \mu c)$).

Proof: For any $\lambda \in \mathbb{R}, \lambda^+ \in \mathbb{R}^+, \# \in \{>, \geq, <, \leq, =, \neq\}$, figure 5 shows how one can build a PCD system of dimension 2 that realizes $x_1 := \lambda^+ x_1$ [1],$x_1 := x_1 + \lambda$ [1], $x_1 := \lambda$ [1], $x_1 := x_1 \# \lambda$ [1] and figure 6 shows how one can build a PCD system of dimension 3 that realizes $x_2 := x_1$ [1] and $x_2 := x_1 + x_2$ [1].

It is easy to transform these PCD systems into PCD systems that realize all the linear machine instructions of definition 3.5: for example to build a PCD system that realizes $x_i := x_j$ [1], take the PCD system $\mathcal{H}' = (X', f')$ of dimension 3 of figure 6 that realizes $x_2 := x_1$ [1], and consider $\mathcal{H} = (\mathbb{R}^{d'}, f)$ where,

[1] That is to say B is an affine basis of V, where V is the minimal affine variety such that $P \subset V$.

[2] We do not assume here that I is necessarily an admissible assignment.

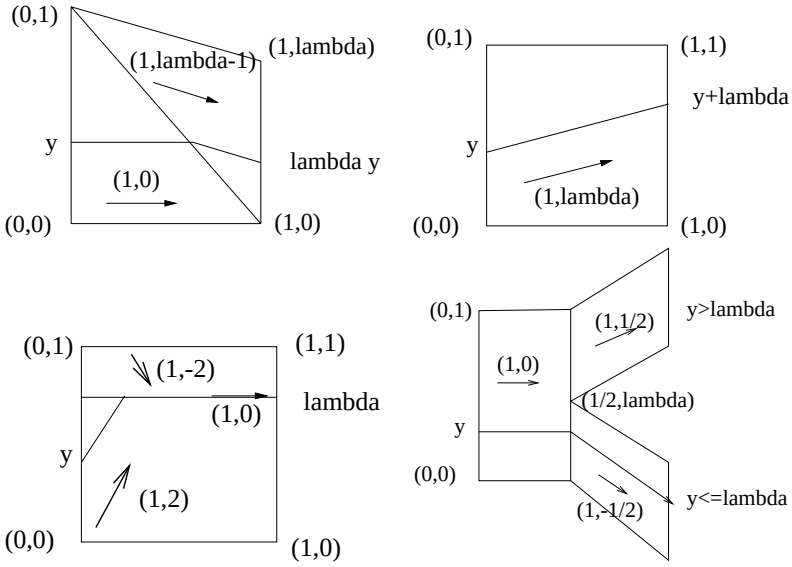


Figure 5: PCD systems realizing $y := \lambda y$ [1], $y := y + \lambda$ [1],$y := \lambda$ [1] and $y > \lambda$? [1].

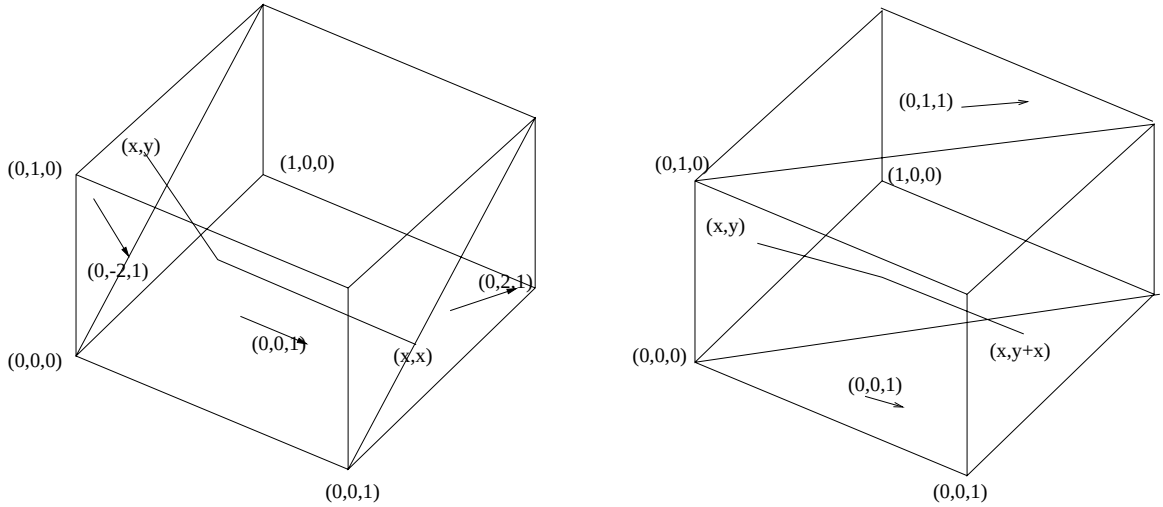


Figure 6: PCD systems realizing $y := x$ [1] and $y := y + x$ [1].

for all $x_1, \ldots, x_{d'}$, $f(x_1, \ldots, x_{d'}) = (x'_1, x'_2, \ldots, x'_{d'})$, $x'_k = 0$ for all $k \notin \{i, j, d+1\}$ and $(x'_j, x'_i, x'_{d+1}) = f'(x_j, x_i, x_{d+1})$.

To realize a linear machine instruction of cost μ instead of 1, multiply all the slopes in the PCD system by $1/\mu$.

□

4.2 Paths and delay module

One can artificially slow down a trajectory (recall that for all $d \in \mathbb{N}$, Id_d denotes the identity function of $[0,1]^d$):

Lemma 4.2 (Delay module) *Let $d \in \mathbb{N}$. Let $d' \geq d+1$ be an integer.*

For any affine function $c : [0,1]^d \to \mathbb{R}^+$, one can build a delay module of time c plus some constant in dimension d'*: for all affine function $c : [0,1]^d \to \mathbb{R}^+$, there exists some $\lambda \in \mathbb{R}^+$, such that one can construct a PCD system of dimension d' that realizes assignment $(Id_d, c + \lambda)$.*

Proof: Take some big enough $k, k' \in \mathbb{R}$ and construct a PCD system as in figure 7.

□

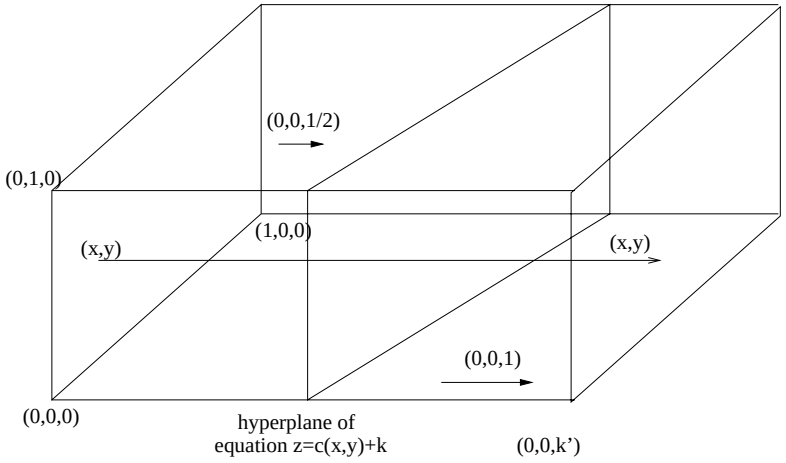


Figure 7: A PCD system realizing a delay

Now, see that one can build some "paths" using the regions of a PCD system: for all $\mu \in \mathbb{R}$, denote abusively by μ the constant function of $[0,1]^d$ whose value is μ: $\forall x \in [0,1]^d, \mu(x) = \mu$.

Lemma 4.3 (Paths) *Let $d \in \mathbb{N}$. Let $d' \geq d+1$ be an integer. Let In and Out be two d-dimensional boxes of $\mathbb{R}^d$.*

For all $\mu \in \mathbb{R}^+$, one can build a path *of time μ between In and Out in dimension d': for all $\mu \in \mathbb{R}^+$, one can build a PCD system of dimension d' that realizes the assignment (Id_d, μ) via input port d-dimensional box In and output port d-dimensional box Out.*

Figure 8: Elementary constructions used in paths: angle and straight path.

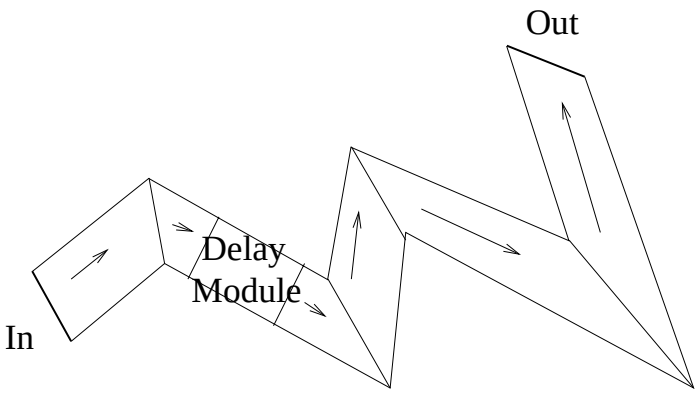


Figure 9: A path between 1-dimensional port In and 1-dimensional port Out.

Proof: Using "angles" and "straight part" as in figure 8 it is easy build some regions that bring any point of coordinates x on In to point of coordinates x on Out. The time taken by a trajectory to go through these regions from point of coordinates x on In to point of coordinates x on Out is some affine function t of x: $t : \mathbb{R}^d \to \mathbb{R}^+$. Using lemma 4.2, insert in one of the regions some regions that realize a delay module of time $-t(x)$ plus some constant. As a consequence, now, the time required by a trajectory to from In to Out is a constant k independent of x. Multiply all the slopes in the regions by μ/k to get a path of time μ: see figure 9.

□

We see now that one can connect several d-dimensional ports to a same d-dimensional port in any dimension $d' \geq d + 2$:

Lemma 4.4 (Merging paths) *Let $d \in \mathbb{N}$. Let $d' \geq d + 2$ be some integer. Let $I_1, \ldots, I_k$ be some d-dimensional boxes. Let Out be a d-dimensional box.*

One can connect all the $I_j, j \in \{1, \ldots, k\}$ to Out in dimension d': there exists a PCD system $\mathcal{H}$ of dimension d' such that, for all $j \in \{1, \ldots, k\}$, $\mathcal{H}$ realizes the assignment $(Id_d, 1)$ via input port I_j and output port Out.

Proof: Using lemma 4.3, for all $j \in \{1, \ldots, k\}$, build a path between I_j and Out: generalize to dimension d the construction of figure 10 to merge all the paths. See that dimension $d + 1$ would not be sufficient to connect the paths.

□

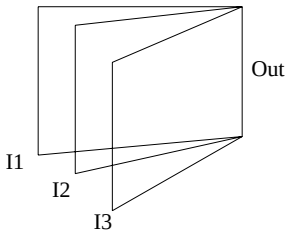


Figure 10: Connecting several 1-dimensional ports I_1, I_2, I_3 to a same 1-dimensional port Out in dimension 3.

4.3 PCD systems can simulate RCT machines

We show now that one can realize all the RCT machines instructions: in particular, one can realize the "Zeno instructions". We use an idea of [2]: see figure 11 and figure 12 to understand how it works. See in figure 13 and in figure 14 how to realize the "special instructions".

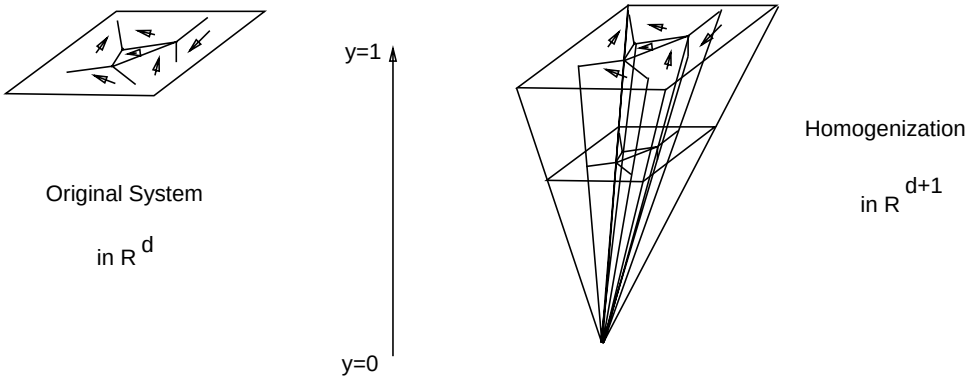


Figure 11: The homogenization of a PCD system of dimension $d = 2$: [2].

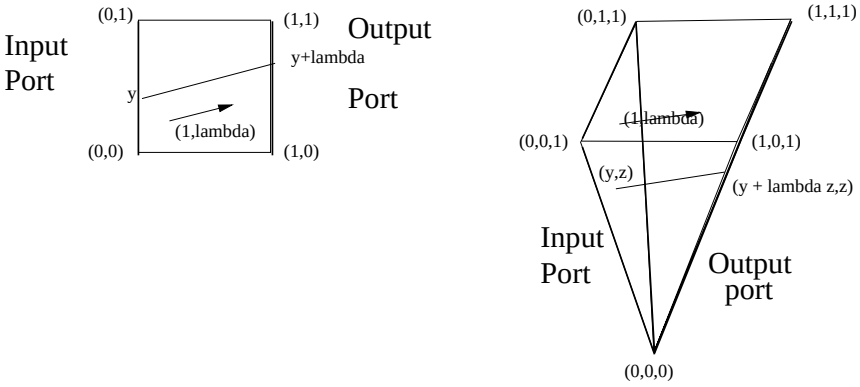


Figure 12: Realizing "Zeno instructions": from a PCD realizing $y := y + \lambda$ [1] of dimension 2 (on left) one can build a PCD system of dimension 3 (on right) that realizes Zeno instruction $(y := y + \lambda$ [1]$)/z$.

Definition 4.1 (Homogenization,Translation) *Let R' be a region of a PCD system of dimension d': that is to say R' is a polyhedral subset of $\mathbb{R}^{d'}$ with some associated slope $s' = (s'_1, \ldots, s'_{d'}) \in \mathbb{R}^{d'}$.*

- *R' is said to be* homogeneous *if the point of of $\mathbb{R}^{d'}$ of coordinates $(0, \ldots, 0)$ is in $\overline{R'}$, where $\overline{R'}$ is the topological closure of R'.*
- *The* translation of region *R' is the region R of $\mathbb{R}^{d'+1}$ defined by:*

$$R = \{(x_1, \ldots, x_{d'+1}) | 0 \leq x_{d'+1} \leq 1 \ \wedge \ (x_1, \ldots, x_{d'}) \in R'\}$$

  with associated slope $s = (s'_1, \ldots, s'_{d'}, 0)$.
- *If $I' = (P', B')$ is a d-dimensional box of $\mathbb{R}^{d'}$, the* translation of *I is the $d+1$-dimensional box defined by $I = (P, B)$, where P is the translation of P', $B = (O, e_1, \ldots, e_{d'+1})$, where $B' = (0, e_1, \ldots, e_{d'})$ and $e_{d'+1}$ is the vector of coordinates $(0, \ldots, 0, 1)$.*
- *The* homogenization of region *R' is the region R of $\mathbb{R}^{d'+1}$ defined by:*

$$R = \{(x_1, \ldots, x_{d'+1}) | 0 < x_{d'+1} \leq 1 \ \wedge \ (x_1/x_{d'+1}, \ldots, x_{d'}/x_{d'+1}) \in R'\}$$

  with associated slope $s = (s'_1, \ldots, s'_{d'}, 0)$.
- *If $I' = (P', B')$ is a d-dimensional box of $\mathbb{R}^{d'}$, the* homogenization of *I is the $d+1$-dimensional box defined by $I = (P, B)$, where P is the homogenization of P', $B = (0, e_1, \ldots, e_{d'+1})$ where $B' = (0', e_1, \ldots, e_{d'})$, O has coordinates $(0, \ldots, 0)$, $e_{d'+1}$ is the vector of coordinates $(o_1, o_2, \ldots, o_{d'}, 1)$ where $(o_1, o_2, \ldots, o_{d'})$ are the coordinates of O' in $\mathbb{R}^{d'}$.*

Lemma 4.5 *Let $\mathcal{H}'$ be a PCD system of dimension d' realizing assignment (f, c) (respectively test (R, c)) of dimension d via input port In and via output port Out (resp. via output ports Out^+, Out^-).*

- *Let $\mathcal{H}$ be the PCD system of dimension $d'+1$ whose regions are the translations of the regions of $\mathcal{H}'$.*

  $\mathcal{H}$ realizes assignment (f, c) (respectively test (R, c)) considered as an instruction of dimension $d+1$ (see definition 3.3) via input port the translation of In and via output port the translation of Out (resp. via output ports the translations of Out^+ and of Out^-).
- *Let $\mathcal{H}$ be the PCD system of dimension $d'+1$ whose regions are the homogenizations of the regions of $\mathcal{H}'$.*

  $\mathcal{H}$ realizes assignment $(f/x_{d+1}, c/x_{d+1})$ (respectively test $(R/x_{d+1}, c/x_{d+1})$) via input port the homogenization of In and via output port the homogenization of Out (resp. via output ports the homogenizations of Out^+ and of Out^-).
- *Let $I = (P, B)$ be a d-dimensional port of $\mathbb{R}^{d'}$. Assume P is homogeneous. If $I' = (P', B')$ and $I'' = (P'', B'')$ denote the translation and the homogenization of I respectively, then $P'' \subset P'$.*

Proof: Immediate from the definitions: see figure 11, figure 12 or see [3].

□

We distinguish a special type of RCT machines:

Definition 4.2 (RCT machine with property *) *Let $M = (Q, q_0, q_f^+, q_f^-, limit^*, \delta)$ be a RCT machine.*

- *A state $q \in Q$ of M is:*
  - flat *[3] iff, for all instantaneous description id of M of type $(q, x_1, \ldots, x_d, t)$, for some $x_1, \ldots, x_d \in [0, 1], t \in \mathbb{R}^+$, for all instantaneous description id' of M with $id' \vdash_d id$, then id' is of type $(q', x'_1, 0, x'_3, \ldots, x'_d, t')$, for some $q' \in Q, x'_1, x'_3, \ldots, x'_d \in [0, 1], t' \in \mathbb{R}^+$.*

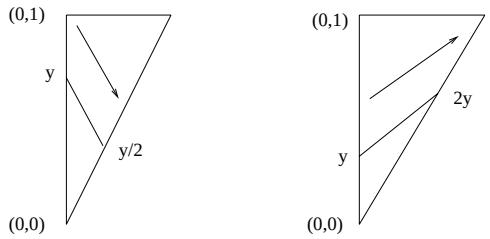


Figure 13: Some PCD systems realizing instruction $x_d := x_d/2[x_d]$ and instruction $x_d := 2x_d[x_d]$.

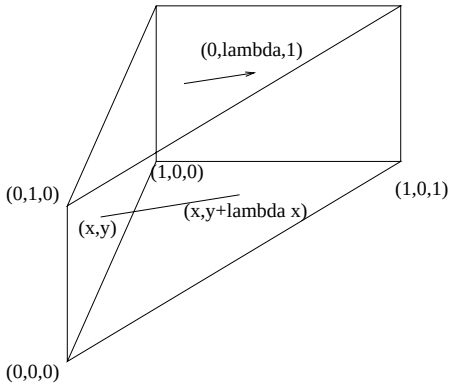


Figure 14: Some PCD system realizing instruction $y := y + \lambda x$ $[x]$.

- separated *[3] iff, for all instantaneous description id of M of type $(q, x_1, \ldots, x_d, t)$, for some $x_1, \ldots, x_d \in [0,1]$, $t \in \mathbb{R}^+$, for all instantaneous descriptions id' and id'' of M of type $(q', x'_1, x'_2, \ldots, x'_d, t')$ and $(q'', x''_1, x''_2, \ldots, x''_d, t'')$ respectively, $q', q'' \in Q, x'_1, \ldots, x'_d, x''_1, \ldots, x''_d \in [0,1]$, $t', t'' \in \mathbb{R}^+$, if $id' \vdash_d id$ and $id'' \vdash_d id$ then $q' = q''$.*

- *M has property $*$ iff all the states $q \in Q$ of M are flat or separated.*

Here is the main theorem of the section: one can simulate a RCT machine by a PCD system:

Theorem 4.1

- *Let M be a RCT machine of dimension d.*

  One can build a PCD system $\mathcal{H}$ of dimension $d+2$ that simulates M.

- *Let M be RCT machine of dimension d with the property *.*

  One can build a PCD system $\mathcal{H}$ of dimension $d+1$ that simulates M.

Proof: Assume M is of dimension d. Assume either that $d' = d+2$ or that $d' = d+1$ and that M has the connectivity property. Denote $M = (Q, q_0, q_f^+, q_f^-, limit^*, \delta)$. For all $q \in Q$, denote $\delta(q) = (q^+, q^-, Instr_q)$.

We prove the theorem by induction over the dimension d and by structural induction over the program of M: we prove that for all M of dimension d one can build a PCD system $\mathcal{H}$ of dimension d': to each state $q \in \mathbb{Q}$ is associated a d-dimensional box I_q and some regions of $\mathcal{H}$ such that $\mathcal{H}$ realizes the assignment (respectively: the test) $Instr_q$ via input port I_q and via output port I_{q^+} (resp. via output ports I_{q^+} and I_{q^-}) using these regions. Moreover, for all $q \in Q$, if $I_q = (P_q, B_q)$, then P_q is homogeneous iff q is not an instruction corresponding to a subprogram of dimension d nor a linear machine instruction. In addition, $\mathcal{H}$ has a box I_* corresponding to the limit state.

Denote $Q' \subset Q$ for the subset of the states of M such that $q \in Q'$ iff $Instr_q$ is either a special instruction, or a Zeno instruction, or obtained from an instruction of dimension $d-1$ which is not a linear machine instruction nor a subprogram of dimension $d-1$.

See that Q' is empty if $d \leq 2$: if $d \leq 2$ skip the five following paragraphs.

Consider $M' = (Q, q'_0, q'^+_f, q'^-_f, limit^*, \delta')$ as a program of dimension $d-1$ where, for all $q \in Q$, $\delta'(q) = (q', q'', Instr')$ iff $q^+, q^- \in Q', q' = q^+, q'' = q^-$, and $Instr' = (f, c)$ if $Instr_q = (f/x_d, c/x_d)$, $Instr' = (R, c)$ if $Instr_q = (R/x_d, c/x_d)$, $Instr' = (Id_{d-1}, 1)$ if $Instr_q$ is a special instruction of type $x_d := x_d/2\ [x_d]$ or of type $x_d := 2x_d[x_d]$, $Instr' = (x_1 := x_1\ [x_k])$ if $Instr_q$ is a special instruction of type $x_d := x_d + \lambda x_k, 2 < k < d$, $Instr' = (f, c)$ if $Instr_q$ is obtained from the instruction (f, c) of dimension $d-1$, and $Instr' = (R, c)$ if $Instr_q$ is obtained from the test (R, c) of dimension $d-1$.

By induction hypothesis one can build a PCD system $\mathcal{H}'$ of dimension $d'-1$ that simulates M'. To each state $q' \in Q'$ of M' corresponds a $d-1$-dimensional port $I'_{q'}$. Moreover, some $d-1$ dimensional box I'_* corresponds to the limit state.

Consider $\mathcal{H}$ as the PCD system built as follows: for all $q \in Q'$, for all region $R' \subset \mathbb{R}^{d'-1}$ of PCD system $\mathcal{H}'$ associated to q:

- if $Instr_q$ corresponds to a special instruction of type $x_d := x_d/2\ [x_d]$ or $x_d := 2x_d[x_d]$, or to a Zeno instruction, then add to $\mathcal{H}$ the homogenization of region R' and take I_q as the homogenization of I'_q.
- if $Instr_q$ corresponds to an instruction obtained from an instruction of dimension $d-1$ or a special instruction of type $x_d := x_d + \lambda x_k, 2 < k < d$ then add to $\mathcal{H}$ the translation of region R' and take I_q as the translation of I'_q.

For each state $q \in Q'$ such that $Instr_q$ is a "special instruction" modify $\mathcal{H}$ as follows: if $Instr_q$ is of type $x_d := x_d/2\ [x_d]$ or of type $x_d := 2x_d\ [x_d]$ we can assume without loss of generality that one region already constructed R of slope s of $\mathcal{H}$ corresponding to state q is the homogenization of a region R' of $\mathcal{H}'$ of slope s' and that R' is of type $R' = A' + [0,1]^{d'-1}$ for some point $A' \in \mathbb{R}^{d'-1}$, where s' is of type $s' = (v, 0, \ldots, 0)$, for some $v \in \mathbb{R}^+$. In that case, replace the slope s of R by $s = (v, 0, \ldots, 0, -v/2)$ if $Instr_q$ is of type $x_d := x_d/2\ [x_d]$ and by $s = (v, 0, \ldots, 0, v)$ if $Instr_q$ is of type $x_d := 2x_d\ [x_d]$. If $Instr_q$ is of type $x_d := x_d + \lambda x_k, 2 < k < d$, we can assume without loss of generality that one region already constructed R of slope s of $\mathcal{H}$ corresponding to state q is the translation of the translation of the translation of the $\ldots$translation of the homogenization of some region R'' of $\mathbb{R}^{d'-d+k-1}$ with slope s'' and that R'' is of type $R'' = A'' + [0,1]^{d'-d+k-1}$ for some point $A'' \in \mathbb{R}^{d'-d+k-1}$, where s'' is of type $s'' = (v, 0, \ldots, 0)$, for some $v \in \mathbb{R}^+$. In that case, replace the slope s of R by $s = (v, 0, \ldots, 0, v)$.

All the ports I_q constructed up to know are either the homogenization of I'_q or the translation of I'_q: but in this latter case, I'_q is always homogeneous. As a consequence, by lemma 4.5, for all $q \in Q'$, it is true that $\mathcal{H}$ realizes the assignment (respectively the test) $Instr_q$ via d-dimensional input port I_q and via d-dimensional output port I^+_q (resp. via output ports I^+_q, I^-_q).

Now, for all $q \in Q, q \notin Q'$ does the following: choose any arbitrary d-dimensional port I_q of $\mathbb{R}^{d'}$ not containing the point of coordinates $(0, \ldots, 0)$. See that $Instr_q$ corresponds to an instruction $Instr_q$ that is either equivalent to a linear machine instruction or either a subprogram of dimension d or $d-1$.

- If $Instr_q$ corresponds to a subprogram of dimension d (respectively: $d-1$), by induction hypothesis, one can build some regions of a PCD system $\mathcal{H}_{instr_q}$ of dimension d' (resp. $d'-1$) that realizes $Instr_q$. Add to $\mathcal{H}$ the regions of $\mathcal{H}_{instr_q}$ (resp. the translation of the regions of $\mathcal{H}_{instr_q}$) and a path of time $1/2$ between the d-dimensional port I_q of $\mathcal{H}$ and the input port of $\mathcal{H}_{instr_q}$ (resp. and the translation of the input port of $\mathcal{H}_{instr_q}$) and a path of time $1/2$ between the output port of $\mathcal{H}_{instr_q}$ (resp. between the translation of the output port of $\mathcal{H}_{instr_q}$) and the d-dimensional port I_{q^+} of $\mathcal{H}$.
- If $Instr_q$ corresponds to a linear machine assignment (f, c) (respectively: to a test (R, c)), by lemma 4.1, build a PCD system $\mathcal{H}_{instr_q}$ of dimension d' that realizes $(f, c/3)$ (resp. $(R, c/3)$). Add to $\mathcal{H}$ the regions of $\mathcal{H}_{instr_q}$ and a path of time $1/3$ between the $d+1$-dimensional port I_q of $\mathcal{H}$ and the input port of $\mathcal{H}_{instr_q}$ and a path (respectively: and two paths) of time $1/3$ between the output port of $\mathcal{H}_{instr}$ and the d-dimensional port I_{q^+} of $\mathcal{H}$ (resp. and the d-dimensional ports I_{q^+} and I_{q^-}).

See that, if $d' = d+2$, by lemma 4.4, all the connections between the ports using the paths can be realized. Now, if $d' = d+1$, there might be some problems of connections between the paths: however, if we assume that M has the property *, when several d- dimensional ports I_{q_1}, I_{q_2} have to be connected to an unique d-dimensional port I_{q^+}, we are sure that this port corresponds to a state $q^+ \in Q$ that is either

separated or flat. If q^+ is separated, there is no problem since the paths connect I_{q_1}, I_{q_2} to different subsets of I_{q^+} [3]. If q^+ is flat, the paths can be taken of dimension $d-1$ by ignoring the value of real register x_2 always equals to 0 [3]: see figure 15.

Define I_* as $\{(x_1, \ldots, x_{d'}) | 0 \leq x_{d'} \leq 1 \ \wedge \ (x_1, \ldots, x_{d'-1}) \in I'_*\}$: I_* is the translation of I'_*. Add a path from port I_* to the port I_{limit^*}.

One gets a PCD system $\mathcal{H}$ that simulates M: $\mathcal{H}$ realizes the assignment corresponding to the execution of M via input port I_{q_0} and via output port $I_{q_f^+}$. Moreover, for all $I_q = (P_q, B_q)$, P_q is homogeneous iff q is not an instruction corresponding to a subprogram of dimension d nor a linear machine instruction. This proves the assertion for dimension d from the assertion in dimension $d-1$.

□

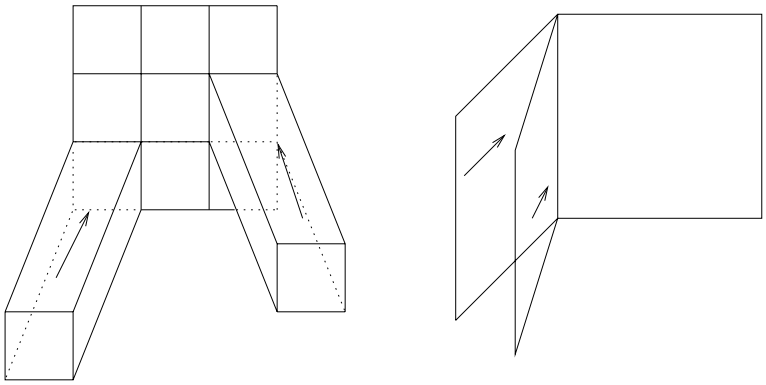


Figure 15: Realizing entrances to input ports of separated (left) and flat (right) states [3].

We claim:

Proposition 4.1 *All the theorems and lemmas proved up to know can be proved using RCT machines with property **

Proof: We say that a RCT program M has the property ** if, in all the IDs of the computations of M, real register x_2 has a value in I. Check that all the programs obtained up to now have the property **.

It is easy to use the trick of claim 17 of [3], to transform any RCT program with the property ** into a RCT program with the property *: transfer all the data from the second real register to the first one when a critical transition must be done: see [3].

□

As a consequence, we get immediately from theorem 4.1 and from theorem 3.5:

Theorem 4.2 *Let $k' \geq 0$.*

- *Any language of $\Delta_{\omega^{k'}}$ can be fully-recognized by a PCD system of dimension $2k'+3$ in finite continuous time.*
- *Any language of $\Delta_{\omega^{k'}+1}$ can be fully-recognized by a PCD system of dimension $2k'+4$ in finite continuous time.*
- *Any language of $\Sigma_{\omega^{k'}}$ can be semi-recognized by a PCD system of dimension $2k'+3$ in finite continuous time.*
- *Any language of $\Sigma_{\omega^{k'}+1}$ can be semi-recognized by a PCD system of dimension $2k'+4$ in finite continuous time.*

5 Upper bounds on the computational power of PCD systems

In this section, we show that theorem 4.2 is optimal. We start by some geometrical considerations.

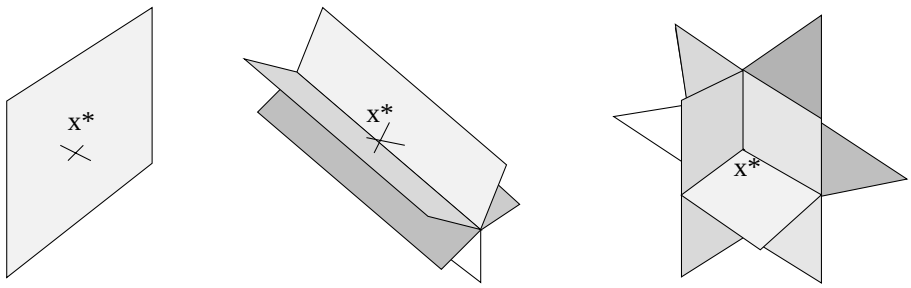


Figure 16: From left to right: x^* is of local dimension $1^+, 2^+, 3$ in a PCD system of dimension 3.

5.1 Geometrical considerations

5.1.1 Local dimension

We define:

Definition 5.1 (Local dimension) *Let $\mathcal{H} = (X, f)$ be a PCD system in dimension d. Let x^* be a point of X. Let Δ be a polyhedral subset $\Delta \subset X$ of maximal dimension $d - d'$ $(1 \leq d' \leq d)$ such that there exists an open convex polyhedron $V \subset X$, with $x^* \in \Delta \cap V$, $\Delta \subset V$, and such that, for any region F of $\mathcal{H}$, $F \cap V \neq \emptyset$ implies $\Delta \subset \overline{F}$ ($\overline{F}$ is the topological closure of F).*

If $d' < d$ then x^* is said to be of local dimension d'^+. *If $d' = d$ then x^* is said to be of* local dimension d' *and we can always choose V small enough such that x^* is the only point of local dimension d' in $\overline{V}$: see figure 16.*

Note that given a rational PCD system $\mathcal{H} = (X, f)$ and $k = d'$ or $k = d'^+$ one can effectively compute $LocDim(\mathcal{H}, k)$ defined as the set of the points $x \in X$ that have a local dimension equal to k.

The idea is that if a point x^* is of local dimension $(d')^+$ in a PCD of dimension d, to study the trajectories in a neighborhood of x^*, one can restrict the attention to a PCD system of dimension d'.

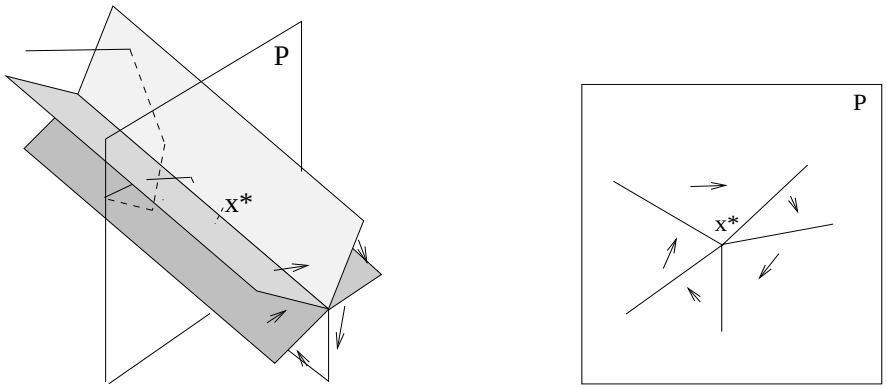


Figure 17: Proposition 5.1: if x^* is of local dimension 2^+ in a PCD $\mathcal{H}$ of dimension 3, the projections on P of the trajectories of $\mathcal{H}$ in a neighborhood V of x^* are the trajectories of a PCD system $\mathcal{H}_{x^*}$ of dimension 2.

Proposition 5.1 *Let $\mathcal{H} = (X, f)$ be a PCD system in dimension d. Let x^* be a point of local dimension $(d')^+$ with $d' < d$. Call P the affine variety of dimension d' which is the orthogonal of Δ in x^*. It is possible to construct a PCD system $\mathcal{H}' = (X' = \mathbb{R}^{d'}, f')$ in dimension d' such that the trajectories of $\mathcal{H}'$ are the orthogonal projections on P of the trajectories of $\mathcal{H}$ in V.*

Proof: Choose an affine basis of $\mathbb{R}^d$ of the form $(x^*, e_1, e_2, \ldots, e_{d'}, \ldots, e_d)$ with $(x^*, e_1, e_2, \ldots, e_{d'})$ taken as a basis of P and $(x^*, e_{d'+1}, \ldots, e_d)$ taken as a basis of Δ. Call $p : \mathbb{R}^d \to \mathbb{R}^{d'}$ the projection that

sends $(x_1, x_2, \ldots, x_d)$ to $(x_1, \ldots, x_{d'})$. By hypothesis, in V the regions are organized as a 'pencil of regions': therefore speed in point $(x_1, x_2, \ldots, x_{d'}, \ldots, x_d) \in V$ does not depend on the coordinates $x_{d'+1}, x_{d'+2}, \ldots, x_d$. The reader can check that $\mathcal{H}' = (X' = \mathbb{R}^{d'}, f')$ where $f'(x_1, x_2, \ldots, x_{d'}) = p(f(x_1, x_2, \ldots, x_{d'}, 0, \ldots, 0))$ is a solution. See figure 17.

□

For any point x^*, the corresponding open convex polyhedron V is denoted by V_{x^*}. $\mathcal{H}'$, Δ are respectively denoted by $\mathcal{H}_{x^*}$ and Δ_{x^*}. If $d' < d$ we denote by p_{x^*} and q_{x^*} the functions that map all point $x \in X$ onto its orthogonal projection on P and onto its orthogonal projection on Δ respectively. If $d' = d$, we define p_{x^*} and q_{x^*} as respectively the identity function and the null function.

A *rational polyhedron* is any polyhedron whose equation can be written using only rational numbers. When $\mathcal{H}$ is a rational PCD system, see that for all point $x^* \in \mathbb{R}^d$, one can always choose V_{x^*} and Δ_{x^*} such that they are rational polyhedral, even if x^* has some non rational coordinates.

We assume the natural order $1 < 1^+ < 2 < 2^+ < \ldots$.

5.1.2 Fundamental properties of points of low or high local dimension

Next lemma is easy:

Lemma 5.1 *For all $d \in \mathbb{N}$, any point of local dimension d of any rational PCD system is a point with rational coordinates.*

Proof: See that if x is a point of local dimension d of some PCD system $\mathcal{H}$ ($\mathcal{H}$ must be of local dimension d), then the intersection of all the regions of $\mathcal{H}$ that intersect V_x is reduced to singleton $\{x\}$. Since all the regions must be rational polyhedral, the unique point of the intersection must have rational coordinates: see figure 16.

□

In [6], we proved:

Lemma 5.2 ([6]) *Let $\mathcal{H} = (X, f)$ be a PCD system of dimension d. Let Φ be a trajectory of $\mathcal{H}$ of finite continuous time T_c and discrete time $T_d \geq \omega$ converging to $x^* = \Phi(T_c)$. Assume that x^* is of local dimension $d' \leq 3^+$. Then necessarily the signature of Φ is ultimately cyclic.*

5.1.3 Trajectories that make some cycles

We define the following relation *Cycle*: see lemmas 5.3 and lemma 5.4 for the motivation.

Definition 5.2 (Relation *Cycle*) *Let d be an integer. Let $\mathcal{H}$ be a PCD system of dimension d. Let z_1, z_2 be two points of $\mathbb{R}^d$. Let $x^* \in \mathbb{Q}^d \cap X$ be a rational point. Let Q be a rational polyhedron.*

We say that $Cycle(z_1, z_2, \mathcal{H}, Q, x^*)$ is true *iff all the following conditions hold simultaneously (see figure 18):*

- *$Q \subset V_{x^*}$, Q is a open convex polyhedron and $z_1, z_2 \in Q$.*
- *$z_1 \neq z_2$, $z_1, z_2 \notin \Delta_{x^*}$ and the line (z_1, z_2) defined by z_1 and z_2 intersects Δ_{x^*} in some point z^*.*
- *$z^* \in \overline{Q}$, where $\overline{Q}$ is the topological closure of polyhedron Q.*
- *$d(p_{x^*}(z_2), p_{x^*}(x^*)) < d(p_{x^*}(z_1), p_{x^*}(x^*))$.*

Recall that d denotes the distance of the maximum. Note that we have always $p_{x^*}(x^*) = x^*$. We prove first that any positive instance of this problem implies that the trajectory is cycling: see figure 18.

Lemma 5.3 *Let $\mathcal{H}$ be a PCD system of dimension d. Let Φ be a trajectory of $\mathcal{H}$. Let $z_1, z_2 \in \mathbb{R}^d$ be two points reached by Φ at time $t_1, t_2 \in \mathbb{R}^+$ respectively with $t_1 < t_2$. Let $x^* \in \mathbb{Q}^d$. Let Q be a rational polyhedron.*

Assume $Cycle(z_1, z_2, \mathcal{H}, Q, x^*)$ is true and that the trajectory stays in Q between time t_1 and time t_2: $\forall t \in [t_1, t_2], \Phi(t) \in Q$.

Then trajectory Φ is cycling and reaches the point z^* of definition 5.2 at time $t^* = t_1 + \sum_{j=0}^{\infty} \lambda^j (t_2 - t_1) = t_1 + (t_2 - t_1) 1/(1-\lambda)$, where $\lambda \in (0,1)$ is such that $d(p_{x^*}(z_2), p_{x^*}(x^*)) = \lambda d(p_{x^*}(z_1), p_{x^*}(x^*))$.

Moreover the trajectory stays in Q between time t_1 and time t^*: for all $t \in [t_1, t^*)$, $\Phi(t) \in Q$.

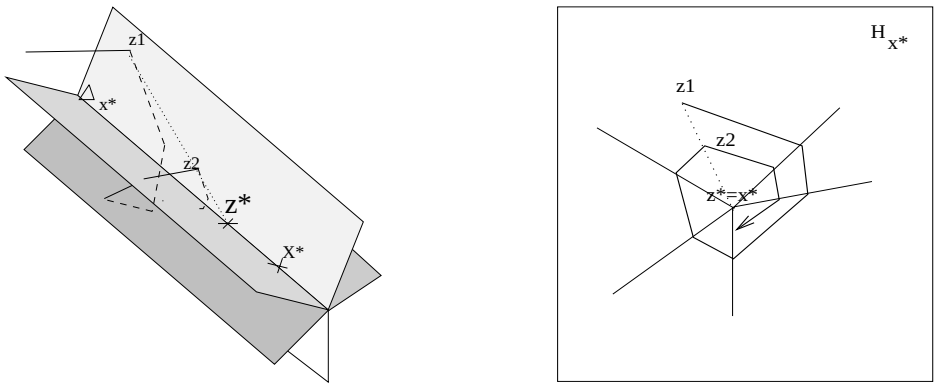


Figure 18: If predicate $Cycle(z_1, z_2, \mathcal{H}, Q, x^*)$ is true for some rational polyhedron Q and some rational point $x^* \in \mathbb{Q}^d$, if the trajectory reaches z_1 and z_2 and does not leave Q between z_1 and z_2, then the trajectory is ultimately cycling and converging to z^*.

When the hypotheses of lemma 5.3 hold, we denote $Cycle_*((t_1, z_1), (t_2, z_2), \mathcal{H}, Q, x^*)$ for the couple $(t^*, z^*) \in \mathbb{R} \times \mathbb{R}^d$.

Proof: Denote $\mathcal{H}_{x^*} = (X', f')$. By lemma 5.1, $\Phi' = p_{x^*}(\Phi)$ must be a trajectory of $\mathcal{H}_{x^*}$. Fix the origin in x^*. $Cycle(z_1, z_2, \mathcal{H}, Q, x^*)$ implies that there exists some real $0 < \lambda < 1$ with $p_{x^*}(z_2) = \lambda p_{x^*}(z_1)$: see figure 18.

By definition of V_{x^*} all the regions of $\mathcal{H}_{x^*}$ intersecting $p_{x^*}(V_{x^*})$ contain $p_{x^*}(x^*)$ in their topological closure. Hence we have $f'(x) = f'(\mu x)$, for all $x \in p_{x^*}(V_{x^*}), \mu \in (0, 1]$. If $\Phi'(t)$ is solution to differential equation $\dot{x}_d = f'(x)$, then $\Psi'(t) = \lambda \Phi'(t/\lambda)$ is also solution. As a consequence, for all $n \geq 2 \in \mathbb{N}$, trajectory Φ' must reach the point $\lambda^{n-1} p_{x^*}(z_1)$ at time $t_1 + (t_2 - t_1) \sum_{j=0}^{n-2} \lambda^j$: see figure 18.

From the definition of $\mathcal{H}_{x^*}$ this implies that, for all $n \geq 2 \in \mathbb{N}$, Φ reaches the point z_n defined by $p_{x^*}(z_n) = \lambda^{n-1} p_{x^*}(z_1)$ and $q_{x^*}(z_n) = q_{x^*}(z_1) + (q_{x^*}(z_2) - q_{x^*}(z_1)) \sum_{j=0}^{n-2} \lambda^j$ at time $t_1 + (t_2 - t_1) \sum_{j=0}^{n-2} \lambda^j$. Hence, trajectory Φ must reach z^* at time t^*: see figure 18. By convexity of Q, Φ must stay in Q between time t_1 and time t^*.

□

5.1.4 Sequence of points of local dimension k

We claim:

Lemma 5.4 *Let $\mathcal{H} = (X, f)$ be a PCD system of dimension d. Let Φ be a trajectory of $\mathcal{H}$: Φ is a function from an interval D of $\mathbb{R}^+$ containing 0 to $\mathbb{R}^d$. Assume that we have a bounded increasing sequence $(t_i)_{i \in \mathbb{N}}$ of real numbers in the domain of function Φ: for all $i \in \mathbb{N}$, $t_i < t_{i+1}$, $t_i \in D$ and there exists some $T \in R$ with $t_i \leq T$ for all $i \in \mathbb{N}$. Denote $t^* = sup_{i \in \mathbb{N}} t_i$.*

- *One can always assume that Φ is defined at time t^*.*
- *$x^* = \Phi(t^*)$ is the limit in $\mathbb{R}^d$ of the sequence $(\Phi(t_i))_{i \in \mathbb{N}}$.*
- *Let $k = (d')$ or $k = (d')^+$ for some integer d'. Assume that for all $i \in \mathbb{N}$, $\Phi(t_i)$ is of local dimension k. Then*
  - *$x^* = \Phi(t^*)$ is of local dimension $> k$.*
  - *If x^* is of local dimension $(d'+1)$ or $(d'+1)^+$, then there must exists $i_1 < i_2 \in \mathbb{N}$, $x^{*'} \in \mathbb{Q}^d$, a rational polyhedron Q such that trajectory Φ stays in Q between time t_{i_1} and time t_{i_2} and such that predicate $Cycle(\Phi(t_{i_1}), \Phi(t_{i_2}), \mathcal{H}, Q, x^{*'})$ is true. Moreover, if $t \in \mathbb{R}, x \in \mathbb{R}^d$ are such that $(t, x) = Cycle_*((t_{i_1}, \Phi(t_{i_1})), (t_{i_2}, \Phi(t_{i_2})), \mathcal{H}, Q, x^*)$ then the local dimension of x is $\geq (d'+1)$.*

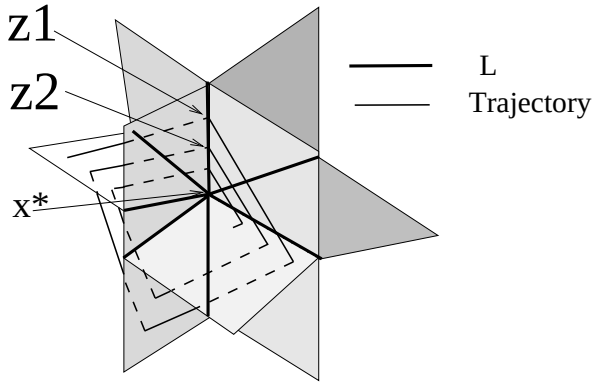


Figure 19: Proof of lemma 5.4: here $d = d' = 3$. $\mathcal{L}$ is the set of the one dimensional regions that intersect $p_{x^*}(V_{x^*})$. $\mathcal{L}$ is made of a finite number of segments. Every time the trajectory reaches a point of local dimension 2^+ , it reaches $\mathcal{L}$. If the trajectory reaches two times $\mathcal{L}$ in a same segment in points z_1, z_2 then predicate $Cycle(z_1, z_2, \mathcal{H}, V_{x^*}, x^{*'})$ is true for all rational point $x^{*'} \in \Delta_{x^*}$.

Proof: By a well-known result of analysis, since Φ is a continuous function and has a bounded right derivative, Φ can always be extended to a function defined on t^*.

Since trajectory Φ is a continuous function, $x^* = \Phi(t^*)$ must be the limit of sequence $(\Phi(t_i))_{i\in\mathbb{N}}$.

Assume that for all $i \in \mathbb{N}$, $\Phi(t_i)$ is of local dimension k. Denote by d'' the local dimension of x^*. By continuity of Φ, there exists $i_0 \in \mathbb{N}$ such that for all $i \geq i_0$, $\Phi(t_i) \in V_{x^*}$. For all $i \geq i_0$, point $\Phi(t_i)$ is of local dimension k and is in V_{x^*}. By considering the dimension of affine subspace $\Delta_{\Phi(t_i)}$, for any $i \geq i_0$, one gets $d'' \geq k$.

Assume $d'' = k$: By some easy geometrical considerations (see figure 16), x^* is the only point of local dimension k in $p_{x^*}(V_{x^*})$. As a consequence, for all $i \geq i_0$, $\Phi(t_i) \in \Delta_{x^*}$. Denote by t_{first} the first point of local dimension $k = d''$ reached by Φ after time t_{i_0}: $t_{first} = \inf\{t | t \in \mathbb{R} \wedge t > t_{i_0} \wedge \Phi(t) \in LocDim(\mathcal{H}, k)\}$. $\Phi' = p_{x^*}(\Phi)$ must be a trajectory of $\mathcal{H}_{x^*}$. Φ does not reach any point of local dimension k at any time t with $t_{i_0} < t < t_{first}$. One has $\Phi'(t_{i_0}) = \Phi'(t_{first})$. As a consequence, for all $n \in \mathbb{N}$, $\Phi'(t_{i_0} + n(t - t_{i_0})) = \Phi'(t_{i_0})$ and all the points of local dimension k reached by Φ at some time $t > t_{i_0}$ must necessarily be reached at some time t of type $t = t_{i_0} + n(t - t_{i_0})$ for some $n \in \mathbb{N}$. In particular, sequence $(t_i)_{i\in\mathbb{N}}$ must be a subsequence of sequence $(t_{i_0} + i(t - t_{i_0}))_{i\in\mathbb{N}}$. We reach a contradiction, since $(t_i)_{i\in\mathbb{N}}$ is assumed to be a bounded sequence. Hence, it is not possible that $d'' = k$ and necessarily $d'' > k$.

Assume $d'' = (d' + 1)$ or $d'' = (d' + 1)^+$. The image $\mathcal{L}$ of $LocDim(\mathcal{H}, k)$ by p_{x^*} is a finite set of one-dimensional segments: see figure 19. Since $(\Phi'(t_i))_{i\geq i_0}$ is an infinite sequence, there must exists some $i_1 < i_2 \in \mathbb{N}$, $z_1 = \Phi(t_{i_1}), z_2 = \Phi(t_{i_2})$ such that $p_{x^*}(z_1)$ and $p_{x^*}(z_2)$ belong to a same segment of $\mathcal{L}$, and such that $d(p_{x^*}(x^*), p_{x^*}(z_2)) < d(p_{x^*}(x^*), p_{x^*}(z_1))$: see figure 19 or figure 18. Take $Q = V_{x^*}$. Check that predicate $Cycle(z_1, z_2, \mathcal{H}, Q, x^{*\prime})$ is then true for any rational point $x^{*\prime} \in \mathbb{Q}^d \cap \Delta_{x^*}$:

Denote $(t, x) = Cycle_*((t_{i_1}, z_1), (t_{i_2}, z_2)), \mathcal{H}, Q, x^*)$. By lemma 5.3, Φ must be converging to x at time t. By definition of $Cycle_*$, x must belong to Δ_{x^*}. As a consequence, x must be of local dimension $\geq (d' + 1)$. □

Corollary 5.1 *Let $\mathcal{H} = (X, f)$ be a PCD system. Let Φ be a trajectory of $\mathcal{H}$ Assume that we have a bounded increasing sequence $(t_i)_{i\in\mathbb{N}}$ of real numbers in the domain of function Φ. Denote $t^* = sup_{i\in\mathbb{N}} t_i$.*

- *For all $d' \in \mathbb{N}$, only a finite number of the points $x_i = \Phi(t_i), i \in \mathbb{N}$ are of local dimension d'.*
- *If the local dimension of $x^* = \Phi(t^*)$ is $k = (d'')$ or $k = (d'')^+$, $d'' \in \mathbb{N}$, then all but a finite number of the $x_i, i \in \mathbb{N}$ are of local dimension $\leq (d'' - 1)^+$.*

Proof: If some x_i is of local dimension d', the dimension d of the PCD system must be equal to d'. If there are a non finite number of points of local dimension d', one can extract from sequence $(t_i)_{i\in\mathbb{N}}$ an infinite sequence $(t'_i)_{i\in\mathbb{N}}$ such that for all $i \in \mathbb{N}$, $\Phi(t'_i)$ is a point of local dimension d' with $t^* = sup_{i\in\mathbb{N}} t'_i$.

This is impossible since by lemma 5.4, x^* must be of local dimension $> d'$ and the dimension of the space must be d'.

By the pigeon hole lemma, since the local dimension is bounded by d the dimension of the space, if the second assertion were false, there must exists some $d \geq d''' > (d'')$ such that one can extract from sequence $(t_i)_{i\in\mathbb{N}}$ an infinite sequence $(t'_i)_{i\in\mathbb{N}}$, such that for all $i \in \mathbb{N}$, $\Phi(t'_i)$ is of local dimension d'''. This is impossible, since by lemma 5.4 one must have $d'' > d'''$.

□

5.2 Some hyper–arithmetical analysis

5.2.1 Representing reals by languages

We represent every point x of $\mathbb{R}^d$, $d \in \mathbb{N}$ by the set of the rational polyhedral that contain x.

Definition 5.3 (Encoding reals by languages) *Let $d \in \mathbb{N}$. Assume that a representation of the rational polyhedral of $\mathbb{R}^d$ over Σ^* is fixed.*

Let $x \in \mathbb{R}^d$. Let $L_x \subset \Sigma^*$ be the language defined as the set of the words $w \in \Sigma^*$ that encodes a rational polyhedron P of $\mathbb{R}^d$ such that $x \in P$. L_x is called the language associated to x. *For all $x \in \mathbb{R}^d$, $d \in \mathbb{N}$, the language L_x associated to x, is denoted by $\lceil x \rceil$.*

We define also:

Definition 5.4 (Encoding real sequences)

- *A* real sequence *is any function h from $\mathbb{N} \times \Sigma^* \times \mathbb{R}^d$ to $\mathbb{R}^d$ for some integers d.*
- *For all $x \in \mathbb{R}^d$, the* relation associated to h corresponding to x *is the langage $R_h(x) \subset \Sigma^*$ defined by $R_h = \{< n, w, P > | n \in \mathbb{N}, w \in \Sigma^*, P \in \Sigma^*$ encodes a rational polyhedron of $\mathbb{R}^d$ such that $P \in \lceil h(n, w, x) \rceil\}$.*
- *For all $k \in \mathbb{N}$, the* relation associated to h corresponding to x up to rank k *is the langage $R_h^{<k}(x) \subset \Sigma^*$ defined by $R_h = \{< n, w, P > | n \in \mathbb{N}, n < k, w \in \Sigma^*, P \in \Sigma^*$ encodes a rational polyhedron of $\mathbb{R}^d$ such that $P \in \lceil h(n, w, x) \rceil\}$.*

5.2.2 Relativizations and the hyper–arithmetical hierarchy

We start by the following lemma:

Lemma 5.5 *There exists a recursive function g such that, for all $y, z \in O$, for all $m \in \mathbb{N}$, if m is an $H(y)$-recursively enumerable index of some set $S \subset \Sigma^*$ and if $y \leq_o z$, then $g(y, m, z)$ is an $H(z)$-recursively enumerable index of S.*

Proof: Denote by h the recursive function of lemma 2.2. Assume $y, z \in O$, $y \leq_o z$. Assume $m \in \mathbb{N}$ is an $H(y)$-recursively enumerable index of set $S \subset \Sigma^*$. S is $H(z)$-recursively enumerable via the machine $M_{m''}^{H(z)}$ with oracle $H(z)$ that on input $w \in \Sigma^*$ simulates $M_m^{H(y)}$, but that replaces any query of machine $M_m^{H(y)}$ of type "$w' \in H(y)$?", $w' \in \Sigma^*$ to its oracle by the query "$\phi_{h(y,z)}(w') \in H(z)$?" to oracle $H(z)$.

The number m'' of this machine depends uniformly in y, m and z and can be given by some recursive function g.

□

Lemma 5.6 (Composition) *Let $X \subset \Sigma^*$.*

- *There exists a recursive g such that, for all $x, y \in O$, $H^{H^X(x)}(y) \leq_m H^X(x +_0 y)$ via $\Phi_{g(x,y)}$.*
- *There exists a recursive h (resp. a recursive h') such that, for all $m, n \in \mathbb{N}, x, y \in O$, if m is some $H^X(x)$-recursively enumerable index of some set $S \subset \Sigma^*$, and if n is some $H^S(y)$-recursively enumerable index of some set $S' \subset \Sigma^*$, then $h(x, y, m, n)$ is an $H^X(x +_0 y)$-recursively enumerable index (resp. $H^X(x +_0 y +_0 2)$-recursive index) of S'.*

Proof: There exists a recursive f, such that for all $n \in \mathbb{N}$, $A, B \subset \Sigma^*$, if $A \leq_m B$ via Φ_n, then $A' \leq_m B'$ via $\phi_{f(n)}$ [16]. Denote by n_0 any integer such that Φ_{n_0} is the identity function.

Let $z, x \in \mathbb{N}$ be given, define ψ by

$$\psi(y) = \begin{cases} n_0 & \text{if } x = 1 \\ f(\phi_z(y)) & \text{if } y = 2^p \text{ for some } p \in \mathbb{N} \\ n' & \text{if } x = 3.5^q, q \in \mathbb{N}, \text{ where } n' \text{ is the number of the} \\ & \text{Turing machine such that } \phi_{n'}(< u, \overline{v} >) =< \phi_{\phi_z(v)}(u), x +_O v) >, \\ & \text{for all } u \in \Sigma^*, v \in O. \\ 0 & \text{otherwise} \end{cases}$$

ψ is partial recursive and an index for ψ can be obtained uniformly from x and z. That is to say, there is a recursive l such that $\psi = \phi_{l(z,x)}$. Applying the fixed point theorem [16], we obtain a recursive function n such that $\phi_{n(x)} = \phi_{l(n(x),x)}$. Take g as $\lambda xy.\phi_{n(x)}(y)$. g is such that for all $x, y \in O$, $H^{H^X(x)}(y) \leq_m H^X(x +_0 y)$ via $\Phi_{g(x,y)}$.

Assume $m \in \mathbb{N}$ is some $H^X(x)$-recursively enumerable index of set $S \subset \Sigma^*$, and $n \in \mathbb{N}$ is some $H^S(y)$-recursively enumerable index of some $S' \subset \Sigma^*$. By proposition 2.1, we have $S \leq_m H^X(2^x) = (H^X(x))'$ via some ϕ_l. As a consequence, $H^S(y) \leq_m H^{H^X(2^x)}(y)$ via some $\phi_{l'}$: see [16] and $H^S(y) \leq_m H^X(2^x +_0 y)$. For all $x, y, 2^x +_0 y \leq_o 2^{x+_0 y}$. Hence, $H^S(y) \leq_m H(2^{x+_0 y})$ via some $\phi_{l''}$ and by proposition 2.1, S' is recursively enumerable in $H^X(x +_0 y)$ with an index n'. Now, see that l, l', ll', n' can be computed effectively from m and n, y and x: see [16]. Hence, n' can be given as a recursive function h of m, n, x, y.

Now, if S' is recursively enumerable in $H^X(x +_o y)$, it is recursive in $H^X(x +_0 y +_o 2)$ with a recursive index computable from any $H^X(x +_o y +_o 2)$ recursively enumerable index of S': this proves the existence of recursive h'.

□

5.2.3 Languages first order definable

We will not distinguish the relations on Σ^* from the languages over Σ^* : a relation R or arity k over Σ^* is considered as the language $\{< n_1, n_2, \ldots, n_k > | R(n_1, \ldots, n_k)\} \subset \Sigma^*$.

Definition 5.5 (First order definition [16]) *Let $Fa_1 \ldots a_n$ a first–order logic expression with free variables $a_1 \ldots a_n$: that is to say $Fa_1 \ldots a_n$ is built up from quantifiers $\exists, \forall$, $=$, sentential connectives $\wedge, \vee, \Rightarrow, \neg$ and relation symbols R_1, R_2, ..., R_k*

Let the relation symbols $R_1, R_2, \ldots, R_k$ be interpreted as certain fixed relations $T_1, \ldots, T_k \subset \Sigma^*$.

Then the relation $R = \{< x_1, \ldots, x_n > | Fa_1 \ldots a_n$ is true over domain Σ^* when $a_1, \ldots, a_n$ are interpreted as $x_1, \ldots, x_n \in \Sigma^*$ respectively and $R_1, R_2, \ldots, R_k$ are interpreted as $T_1, \ldots, T_k \subset \Sigma^*$ respectively $\} \subset \Sigma^*$ is said to be definable by first order formula F from relations $T_1, \ldots, T_k$.

In a first–order logic expression, the quantifications over functions are not allowed. All the quantifications are on variables. Here, the variables are interpreted as words of Σ^*. As an example, if $T \subset \Sigma^*$ is some binary relation, then $\{n \in \Sigma^* | \exists t \in \Sigma^*\ T(n,t)$ *is true*$\}$ is first order definable by formula $\exists t\ R(n,t)$ from relation T.

Proposition 5.2 (Tarski-Kuratowski algorithm [16])

- *Let F be a first order formula. F can always be transformed into a first order formula in prenex form logically equivalent to F beginning with a quantifier $\exists$.*
- *Assume F is a first order formula in prenex form beginning with a quantifier $\exists$. Let $n \in N$ be the number of quantifier alternations[3] in formula F.*
  - *Let $R \subset \Sigma^*$ be a language defined by formula F from some recursive relations $T_1, \ldots, T_k$ (respectively: defined by formula F from some A-recursive relations $T_1, \ldots, T_k$).*

    Then R is in the arithmetical hierarchy (resp. in the A-arithmetical hierarchy): $R \in \Sigma_{n+1}$ (resp. $R \in \Sigma^A_{n+1}$).

[3] The number of alternations is the number of pairs of adjacent but unlike quantifiers in the prefix of the prenex formula[16].

– *The dependence of R on relations $T_1, \ldots, T_k$ is uniform: assume first order formula F is fixed. There exists a recursive g_F, such that, for all $n_1, \ldots, n_k \in \mathbb{N}$ (resp. for all $n_1, \ldots, n_k \in \mathbb{N}$, for all $A \subset \Sigma^*$), if $n_1, \ldots, n_k$ are recursive (respectively: A-recursive) indexes of relations $T_1, \ldots, T_k$ respectively, then $g_F(n_1, \ldots, n_k)$ is an $H(y)$-recursively enumerable index (resp. is an $H^A(y)$-recursively enumerable index) of language R defined by formula F from relations $T_1, \ldots, T_k$.*

5.3 Sampling a PCD system up to local dimension 3^+

5.3.1 Linear machines and affine maps

Definition 5.6 (Linear machines [13]) *A* rational linear machine M of dimension d *is a finite dimensional linear machine of [13] whose constants are in $\mathbb{Q}$: see [13] for a formal definition.*

A computation of M on discrete input $w \in \Sigma^*$ and on continuous input $x = (x_1, \ldots, x_{d'}) \in \mathbb{R}^{d'}$, $d' < d$, is a computation of M starting from $(\mathcal{I}(w), x_1, \ldots, x_{d'}, 0, \ldots, 0)$.

Here is an informal definition: a rational linear machine is a RCT machine of dimension d whose program is made only of the "linear machine instructions" (see definition 3.5) and whose real registers are not restricted to have a value in $[0, 1]$ but can have any value of $\mathbb{R}$. Thus, in a rational linear machine the real registers x_i can have any value of $\mathbb{R}$, all the instructions have cost 1, and all the instructions are of type $x_i := x_i + x_k$, $x_i := x_j$, $x_i := \lambda x_i$,$x_i := \lambda$, $x_i := x_i + \lambda$, $x_i \# \lambda$, where $\# \in \{>, \geq, <, \leq, =, \neq\}$ and $\lambda \in \mathbb{Q}$.

A *rational affine map* is any affine map $h : \mathbb{R}^d \to \mathbb{R}^d$ of type $h(x) = Ax + b$ for some rational matrices A, B. We prove:

Lemma 5.7 *Let $h : \mathbb{R}^d \to \mathbb{R}^d$ be a rational affine map: $h(x) = Ax + B$ for some rational matrices A and B.*

There exists some $n_h \in \mathbb{N}$ such that for all $x \in \mathbb{R}^d$, $\lceil h(x) \rceil \leq_m \lceil x \rceil$ via ϕ_{n_h}.

Proof: We have $h(x) \in P$, for some polyhedron P, iff $x \in h^{-1}(P)$. For all rational polyhedron P, $h^{-1}(P)$ is a computable rational polyhedron. Hence $\lceil h(x) \rceil \leq_m \lceil x \rceil$ via the recursive function ϕ_{n_h} of number n_h that maps any encoding of a polyhedron P of $\mathbb{R}^d$ to an encoding of polyhedron $h^{-1}(P)$ of $\mathbb{R}^d$.

□

5.3.2 Rational and Purely rational PCD systems

In [6] the following lemma is proved:

Lemma 5.8 *One can build a rational linear machine M such that:*

- *if M is given as discrete input a rational PCD system $\mathcal{H} = (X, f)$ of dimension d, a finite sequence of distinct regions $(F_0, F_1, \ldots, F_j)$ of $\mathcal{H}$ and as continuous input a point $x \in \mathbb{R}^d$, then M answers the following question:*

  "Does the trajectory Φ starting from x have a periodic signature of type $(F_0, F_1, \ldots, F_j)^\omega$ and then reach a point $x^* \in X$ of local dimension $\leq 3^+$ at a finite continuous time t^*?"

- *whenever the answer is positive, M outputs x^* and t^*.*

Proof: See that the proof given in [6] can be transformed easily into a linear machine algorithm, using lemma 3.1 and the technics of [10] to simulate any arbitrary division or multiplication of some real register by some computable rational number. This is clear for most of the instructions of the algorithm given in [6] except, may be, for the instructions of type $x_i^* = x_i^0 + \Sigma_{j=0}^{\infty} Off_i(x_1^j, x_{d_0'}^j)$: see the notations of [6]. But, check that for any linear map A from $\mathbb{R}^2$ to $\mathbb{R}^2$ (resp. $\mathbb{R}$ to $\mathbb{R}$) with a rational matrix, the linear map $\Sigma_{j=0}^{\infty} A^j$, when it exists, is always a linear map ΣA from $\mathbb{R}^2$ to $\mathbb{R}^2$ (resp. $\mathbb{R}$ to $\mathbb{R}$) with a rational matrix whose coefficients are computable from the coefficients of A. As a consequence, the instruction $x_i^* = x_i^0 + \Sigma_{j=0}^{\infty} Off_i(x_1^j, x_{d_0'}^j)$ can be simulated by computing the rational matrix of ΣA and by replacing this instruction by $x_i^* = x_i^0 + Off_i(\Sigma A(x_1^0, x_2^0))$: see the proof in [6].

□

Theorem 5.1 • *Any rational PCD system $\mathcal{H}$ of dimension $d \leq 4$ is purely rational.*

• *There exists a rational PCD system of dimension 5 that is not purely rational.*

Proof: Let Φ be a trajectory of a PCD system of dimension $d \leq 4$ starting from a rational point. Assume Φ enters a region in a point $x = \Phi(t)$, where x has some non-rational coordinates, and x is the first point reached by Φ with this property. By lemma 5.1, x must be of local dimension $(d')^+$ for some d'. We must have $d' \leq 3$. By lemma 5.2, from some time $t_0 < t$ up to time t, the signature of the restriction of Φ to $[t_0, t)$ must be cyclic of type $(F_0, F_1, \ldots, F_j)^\omega$. Let $t' \in (t_0, t)$ with $\Phi(t') \in F_0$. By lemma 5.8, a rational linear machine M with discrete input $\mathcal{H}, (F_0, F_1, \ldots, F_j)$ and continuous input x' outputs x: it is clear that if a rational linear machine is started with all its continuous inputs in $\mathbb{Q}$, then any value output by the machine is in $\mathbb{Q}$. Hence, x cannot have some non rational coordinates.

Now, the second assertion is immediate from theorem 4.2 that proves that one can recognize some non-arithmetical sets in dimension 5 and from [6] that proves that any set recognized by a purely rational PCD system is arithmetical.

□

5.3.3 Sampling a PCD system up to local dimension 3^+ using linear machines

Denote by $\mathcal{P} \subset \Sigma^*$ the set of the rational polyhedron of $\mathbb{R}^d$. We define the sampling of a trajectory: see lemma 5.9 for the motivation.

Definition 5.7 (Sampling of a PCD system) *Assume a rational PCD system $\mathcal{H} = (X, f)$ of dimension d is fixed.*

- *A* sampling of $\mathcal{H}$ *is a mapping g from $\mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ with the following properties: assume $Q \in \mathcal{P}$, $t \in \mathbb{R}$, $x \in \mathbb{R}^d$ are fixed. Denote by Φ the trajectory of $\mathcal{H}$ starting from x at time t.*
  - *For all $k \in \mathbb{N}$, $g(k, Q, t, x)$ is a couple (t_k, x_k), $t_k \in \mathbb{R}, x_k \in \mathbb{R}^d$ such that $\Phi_x(t_k) = x_k$.*
  - *$t_0 = t, x_0 = x$.*
  - *$t_{k+1} \geq t_k$ for all $k \in \mathbb{N}$.*
  - *Only one of two following cases hold:*
    - *there is some $k_0 \in \mathbb{N}$ with $x_{k_0} \in Q \cup NoEvolution(\mathcal{H})$, and for all $k \geq k_0$, $x_k = x_{k_0}, t_k = t_{k_0}$.*
    - *$t_k < t_{k+1}$ for all $k \in \mathbb{N}$ and Φ does not reach $Q \cup NoEvolution(\mathcal{H})$ at any time $t < sup_{k \in \mathbb{N}} t_k$.*
- *If there exists a rational number $t_{sup} \in \mathbb{Q}$ such that $t_k \leq t_{sup}, t_k < t_{k+1}$ for all $k \in \mathbb{N}$, then sampling g is said to be* Zeno for Q, t and x. *By lemma 5.4, when g is Zeno for Q, t and x, the sequence $(x_k)_{k \in \mathbb{N}}$ is converging to some $x^* = \Phi(t^*)$, where $t^* = sup_{k \in \mathbb{N}} t_k$.*
- *Sampling g is said to be a* sampling up to local dimension l, *where $l = d'$ or $l = (d')^+$ for some integer $d' \leq d$, if for all $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, when g is Zeno for Q, t, x, then the point reached at time $t^* = sup_{k \in \mathbb{N}} t_k$ has a local dimension $> l$.*

Note that a sampling is a real sequence. See that there exists a fixed first order formula F such that, for all $t \in \mathbb{R}, x \in \mathbb{R}^d$, the language $\{Q | g$ is Zeno for $Q, t, x\}$ is first order definable by formula F from relation $R_g(t, x)$: this formula F is $\exists t_{sup} \in \mathbb{Q}\ \forall k \in \mathbb{N}\ t_k \leq t_{sup}\ \wedge \forall k x_k \notin Q \cup NoEvolution(\mathcal{H})$.

Here is a restatement of a lemma of [6]:

Lemma 5.9 *Let $\mathcal{H}$ be a rational PCD system.*

One can build a rational linear machine M that computes a sampling of $\mathcal{H}$ up to local dimension 3^+: there exists a sampling $g : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ of $\mathcal{H}$ and a rational linear machine M that on discrete input $< k, Q >$, $k \in \mathbb{N}, Q \in \mathcal{P}$ and on real inputs $t \in \mathbb{R}, x \in \mathbb{R}^d$ outputs $g(k, Q, t, x)$.

Proof: On discrete input $< k, Q >, k \in \mathbb{N}, Q \in \mathcal{P}$ and on continuous inputs $t \in \mathbb{R}, x \in \mathbb{R}^d$ M starts by subdividing the regions of $\mathcal{H}$ if necessary, so that Q (respectively: $NoEvolution(\mathcal{H})$) is a finite union of regions of $\mathcal{H}$: denote by m the number of regions of the resulting PCD system. Then M evolves according to the following algorithm:

Algorithm 17 Program M

$t_0 = t, x_0 = x, Stop_0 = False$
for $i = 0$ **to** $k - 1$ **do**
  if $Stop_i$ is true **then**
    Set $t_{i+1} := t_i, x_{i+1} := x_i, Stop_{i+1} := True$.
  else
    Determine the first $m + 1$ regions $F^0, F^2, \ldots, F^m$ of the signature of the trajectory starting from x_i.
    Take j, if it exists, as the least integer such that $F^0 = F^j$.
    if j exists and all the regions $F^0, \ldots, F^{j-1}$ are not in $Q \cup NoEvolution(\mathcal{H})$
    then
      Using the Turing linear machine of lemma 5.8, test if the trajectory starting from x_i has a periodic signature of type $(F^0, F^1, \ldots, F^{j-1})^\omega$ and then reach some point x^* of local dimension $\leq 3^+$ at some finite continuous time t^*.
      if it is so **then**
        Compute t^* and x^* using lemma 5.8.
        Set $x_{i+1} := x^*, t_{i+1} := t^*$
      end if
    end if
  end if
  if x_{i+1}, t_{i+1} have not received a value yet **then**
    Compute the point x and the time t respectively of the next intersection of the trajectory of $\mathcal{H}$ starting from x_i at time t_i with some region of $\mathcal{H}$
    Set $x_{i+1} := x, t_{i+1} := t$.
  end if
  if $x_{i+1} \in Q \cup NoEvolution(\mathcal{H})$ **then**
    Set $Stop_{i+1} := True$
  else
    Set $Stop_{i+1} := False$.
  end if
end for
Output t_k and x_k.

This algorithm is clearly a rational linear machine algorithm. Denote by Φ the trajectory starting from x at time t. All the t_k, x_k output are such that $\Phi(t_k) = x_k$. If there is some $k_0 \in \mathbb{N}$ with $x_{k_0} \in Q \cup NoEvolution(\mathcal{H})$, then $x_k = x_{k_0}$ for all $k \geq k_0$. If for all $k \in \mathbb{N}$ $x_k \notin Q \cup NoEvolution(\mathcal{H})$, then it clear then $t_k < t_{k+1}$ for all $k \in \mathbb{N}$ and it is easy to see that Φ does not reach $Q \cup NoEvolution(\mathcal{H})$ at any time $t < t^* = sup_{k \in \mathbb{N}} t_k$. Assume now, that g is Zeno for Q, t, x: denote $x^* = \Phi(t^*)$. Assume x^* is of local dimension $\leq 3^+$: by lemma 5.2 there must exist some $t_k \leq t^*$ such that the trajectory starting from x_k has a periodic signature of type $(F^0, F^1, \ldots, F^{j-1})^\omega$. By this algorithm, we must have $t_{k+1} = t^*, x_{k+1} = x^*$. This is in contradiction with the definition of t^* as $t^* = sup_{k \in \mathbb{N}} t_k$, since it implies that $t_{k+2} > t_{k+1} = t^*$.

□

5.3.4 Sampling a PCD system up to local dimension 3^+ using Turing machines

We show that any rational linear machine can be simulated by a Turing machine with oracle:

Lemma 5.10 *Let M be a rational linear machine of dimension k, $k \in \mathbb{N}$.*

Assume M computes some function $f_M : \Sigma^* \times \mathbb{R}^d \to \mathbb{R}^d$: for all $w \in \Sigma^*$, for all $x \in \mathbb{R}^d$, M started with discrete input w and continuous input x outputs $f_M(w,x) \in \mathbb{R}^d$.

There exists a recursive $h_M : \Sigma^* \to \mathbb{N}$ such that for all $w \in \Sigma^*$, for all $x \in \mathbb{R}^d$, $W^{\lceil x \rceil}_{h_M(w)} = \lceil f_M(w,x) \rceil$.

Proof:

Assume $w \in \Sigma^*$ and $x \in \mathbb{R}^d$ are momentarily fixed.

We build a Turing machine M' with oracle $\lceil x \rceil$ that, on input $w' \in \Sigma^*$, simulates M on discrete input $w \in \Sigma^*$ and real input $x \in \mathbb{R}^d$ until M accepts and then accepts iff $w' \in \lceil f_M(w,x) \rceil$.

For all $t \in \mathbb{N}$, denote by $(q^t, x_1^t, \ldots, x_k^t, t)$ the ID of M at time $t \in \mathbb{N}$ on discrete input w and real input x. Denote $x^t = (x_1^t, \ldots, x_k^t) \in \mathbb{R}^k$. M' simulates M as follows: at time t, M' has $q^t, t, n_t \in \mathbb{N}$ on its tape where $\lceil x^t \rceil \leq_m \lceil x \rceil$ via ϕ_{n_t}.

Check that simulating a test of M at time t is equivalent to answer the question "is $(x_1^t, x_2, \ldots, x_k^t)$ in P ?" for some rational polyhedron P of $\mathbb{R}^k$. As a consequence, it can be simulated by M' by the query "$\phi_{n_t}(P) \in \lceil x \rceil$?" to oracle $\lceil x \rceil$. M' sets $n_{t+1} = n_t$ and sets q_{t+1} according to the answer of the query.

Check that simulating an assignment of M at time t is equivalent to doing $(x_1^{t+1}, \ldots, x_k^{t+1}) = h(x_1^t, \ldots, x_k^t)$ for some rational affine map $h : \mathbb{R}^k \to \mathbb{R}^k$. As a consequence, it can be simulated by M' by setting q^{t+1} to the state corresponding to the next instruction of M and by setting n_{t+1} to the number of the Turing machine such that $\phi_{n_{t+1}} = \phi_{n_t}(\phi_{n_h})$, where n_h is the number given by lemma 5.7 applied on mapping h.

Hence, M' simulates all the instructions of M until M accepts at some time $t \in \mathbb{N}$. Then M' determines if $w' \in \lceil f_M(w,x) \rceil$ by making the query "$\phi_{n_t}(w') \in \lceil x \rceil$?".

This gives a Turing machine M' with oracle $\lceil x \rceil$ that recognizes $\lceil f_M(w,x) \rceil$. One can easily compute the number of this machine M': this number is independent of x and can be given as a recursive function h_M of $w \in \Sigma^*$.

□

We get:

Corollary 5.2 *Let $\mathcal{H}$ be a rational PCD system.*

There exists a sampling $g : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ of $\mathcal{H}$ up to local dimension 3^+ and $n \in \mathbb{N}$, such that, for all $t \in \mathbb{R}$, for all $x \in \mathbb{R}^d$, $W_n^{\lceil (t,x) \rceil} = R_g(t,x)$, where R_g is the relation associated to g corresponding to (t,x).

Proof: By lemma 5.9, there exists a sampling $g : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ of $\mathcal{H}$ up to local dimension 3^+ and a rational linear machine M that on input $< k, Q >$, $k \in \mathbb{N}, Q \in \mathcal{P}$ and on real inputs $t \in \mathbb{R}, x \in \mathbb{R}^d$ outputs $g(k,Q,t,x)$. By lemma 5.10, one can find a recursive h_M, such that for all $k \in \mathbb{N}, Q \in \mathcal{P}$ for all $t \in \mathbb{R}, x \in \mathbb{R}^d$, $W^{\lceil (x,t) \rceil}_{h_M(<k,Q>)} = \lceil g(k,Q,t,x) \rceil$. Take n as the number of the Turing machine with oracle that on input $< k, Q, P >$ simulates the machine with oracle of number $h_M(< k, Q >)$ on input P.

□

5.4 Sampling a PCD system up to local dimension d

We fix in this subsection a PCD system $\mathcal{H}$ of dimension d.

5.4.1 HyperJump operation

Definition 5.8 (HyperJump operation) *Assume we have a sampling g of $\mathcal{H}$.*

We define $HyperJump[g] : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ as follows: assume $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ are fixed.

- *Set $HyperJump[g](0,Q,t,x) = (t,x)$*
- *Let $k \geq 1$. Denote $HyperJump[g](k-1,Q,t,x) = (t_{k-1}, x_{k-1}), t_{k-1} \in \mathbb{R}, x_{k-1} \in \mathbb{R}^d$.*
  - *If g is Zeno for Q, t_{k-1}, x_{k-1} or if there exists some $k_0 \in \mathbb{N}$ such that $x'_{k_0} \in Q \cup NoEvolution(\mathcal{H})$ where $g(k_0, Q, t_{k-1}, x_{k-1}) = (t'_{k_0}, x'_{k_0})$, then set $HyperJump[g](k,Q,t,x)$ as the limit of the sequence $(g(k', Q, t_{k-1}, x_{k-1}))_{k' \in \mathbb{N}}$*

– *Otherwise, set* $HyperJump[g](k,Q,t,x) = g(k,Q,t_{k-1},x_{k-1})$

Lemma 5.11 *Assume we have a sampling g of $\mathcal{H}$ up to local dimension $(d')^+$ for some integer d'. Then:*

- *$HyperJump[g]$ is a sampling of $\mathcal{H}$ up to local dimension $(d'+1)^+$.*
- *Assume $HyperJump[g]$ is Zeno for some $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$. Then for all $k \in \mathbb{N}$, x_k is of local dimension $\geq (d'+1)$, where $HyperJump[g]\ (k,Q,t,x) = (t_k,x_k)$, $t_k \in \mathbb{R}, x_k \in \mathbb{R}^d$.*
- *For all $t \in \mathbb{R}$, for all $x \in \mathbb{R}^d$, denote by $R_g(t,x)$ the relation associated to real sequence g corresponding to (t,x).*

  There exists a fixed first order formula F such that for all $k \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, $\lceil HyperJump[g](k+1,Q,t,x)\rceil$ is definable by formula F from relation $R_g(HyperJump[g](k,Q,t,x))$ and from some recursive relations.

Proof: By the remark page 42, there exists a fixed first order formula over relation $R_g(t,x)$ that tells if g is Zeno for Q,t,x. There exists clearly a fixed first order formula G such that for all real sequence $(g'(k',Q,x))_{k' \in \mathbb{N}}$ converging to some $g'^*(Q,x) \in \mathbb{R}^d$, $\lceil g'(Q,x)\rceil$ is definable by formula G from relation $R_{g'}(x)$, where $R_{g'}$ denotes the relation associated to g' corresponding to x. As a consequence, the last assertion is clear, since definition 5.8 can be translated directly into a fixed first order formula F that, for all k,Q,t,x, defines $\lceil HyperJump[g](k+1,Q,t,x)\rceil$ from some recursive relations and from relation $R_g(HyperJump[g](k,Q,t,x))$.

We prove now that $HyperJump[g]$ is a sampling of $\mathcal{H}$ up to local dimension $(d'+1)^+$.

Assume $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ are fixed. Denote $HyperJump[g](k,Q,t,x) = (t_k,x_k)$, $t_k \in \mathbb{R}, x_k \in \mathbb{R}^d$, for all $k \in \mathbb{N}$. Let Φ be the trajectory of $\mathcal{H}$ starting from x at time t.

From the fact that g is a sampling it is easy to show by induction over k that for all $k \in \mathbb{N}\, \Phi(t_k) = x_k$. Now, if there is some k_0 with $x_{k_0} \in Q \cup NoEvolution(\mathcal{H})$, since g is a sampling, it is clear than $x_k = x_{k_0}, t_k = t_{k_0}$ for all $k \geq k_0$. If for all $k, x_k \notin Q \cup NoEvolution(\mathcal{H})$, it is easy to see that $t_{k+1} > t_k$ for all $k \in \mathbb{N}$. Hence $HyperJump[g]$ is a sampling.

Assume that $HyperJump[g]$ is Zeno for some Q,t,x. For all $k \in \mathbb{N}$, g must be Zeno for Q,t_{k-1},x_{k-1}: hence, (t_k,x_k) is the limit of $g(k',Q,t_{k-1},x_{k-1}), k' \in \mathbb{N}$. Since g is a sampling up to local dimension $(d')^+$, the local dimension of x_k must be $> (d')^+$ for all $k \in \mathbb{N}$. This proves the second assertion.

Denote $t^* = sup_{k \in \mathbb{N}} t_k$ and $x^* = \Phi(t^*)$. By lemma 5.4, the local dimension of x^* is $> (d'+1)^+$. This proves the first assertion.

□

5.4.2 CycleFree operation

Definition 5.9 (Cycle Free operation) *Assume we have a sampling g of $\mathcal{H}$.*

We define $CycleFree[g] : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ as follows: assume $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ are fixed.

- *Set $CycleFree[g](0,Q,t,x) = (t,x)$*
- *Let $k \geq 1$. Denote $CycleFree[g]\ (k-1,Q,t,x) = (t_{k-1},x_{k-1})$, $t_{k-1} \in \mathbb{R}, x_{k-1} \in \mathbb{R}^d$.*
  - *Either there exists $k_2 \in \mathbb{N}$, $k_2 < k$, some $x^* \in \mathbb{Q}^d$, a rational polyhedron F not intersecting Q, such that $Cycle(x_{k-1},z_2,\mathcal{H},F,x^*)$ is true, $x_{k-1} \notin Q, z_2 \notin Q$, $z_2 \notin F^c$, where F^c is the complement of polyhedron F on $\mathbb{R}^d$ and $HyperJump[g](k_2,Q \cup F^c,t_{k-1},x_{k-1}) = (t_2,z_2)$.*

    Set $CycleFree[g](k,Q,t,x) = Cycle_*((t_{k-1},x_{k-1}),(t_2,z_2),\mathcal{H},F,x^*)$
  - *or this is false:*

    Set $CycleFree[g](k,Q,t,x) = HyperJump[g](k,Q,t_{k-1},x_{k-1})$

Lemma 5.12 *Assume g is a sampling of $\mathcal{H}$ up to local dimension $(d')^+$ for some $d' \in \mathbb{N}$. Then:*

- *$CycleFree[g]$ is a sampling of $\mathcal{H}$ up to local dimension $(d'+2)^+$.*
- *For all $k \in \mathbb{N}$, $t \in \mathbb{R}, x \in \mathbb{R}^d$, denote by $R^{<k}_{HyperJump[g]}(t,x)$ the relation associated to real sequence $HyperJump[g]$ corresponding to (t,x) up to rank k.*

  There exists a fixed first order formula F such that for all $k \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, $\lceil CycleFree[g](k+1,Q,t,x)\rceil$ is definable by formula F from some recursive relations and from relation $R^{<k}_{HyperJump[g]}(CycleFree[g](k,Q,t,x))$.

Proof: It is easy to see that there exists a fixed first order formula G such that, for all $z_1, z_2 \in \mathbb{R}^d$, $\{< Q, x^* > | Q \in \mathcal{P}, x^* \in \mathbb{Q}^d, Cycle(z_1, z_2, \mathcal{H}, Q, x^*)$ is true $\}$ is definable by formula G from relations $\lceil z_1\rceil, \lceil z_2\rceil$ and from some recursive relations. Now, see that there also exists a fixed first order formula H such that $\lceil Cycle_*((t_1,z_1),(t_2,z_2),\mathcal{H},Q,x^*)\rceil$ is defined by formula H from relations $\lceil (t_1,z_1)\rceil$, $\lceil (t_2,z_2)\rceil$ and from some recursive relations. As a consequence, definition 5.4 can be translated directly into a fixed first order formula F such that, for all $k \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, $\lceil CycleFree[g](k+1,Q,t,x)\rceil$ is definable by formula F from relation $R^{<k}_{HyperJump[g]}(CycleFree[g](k,Q,t,x))$ and from some recursive relations. This proves the second assertion.

We prove now that $CycleFree[g]$ is a sampling of $\mathcal{H}$ up to local dimension $(d'+2)^+$. Assume $Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ are fixed. Denote $CycleFree[g](k,Q,t,x) = (t_k, x_k)$, $t_k \in \mathbb{R}, x_k \in \mathbb{R}^d$, for all $k \in \mathbb{N}$. Let Φ be the trajectory of $\mathcal{H}$ starting from x at time t.

Using lemma 5.11 and lemma 5.3, it is easy to show by induction over k that for all k, $\Phi(t_k) = x_k$. If there is some k_0 with $x_{k_0} \in Q \cup NoEvolution(\mathcal{H})$, since $HyperJump[g]$ is a sampling, it is clear than $x_k = x_{k_0}, t_k = t_{k_0}$ for all $k \geq k_0$. If for all $k \in \mathbb{N}$, $x_k \notin Q \cup NoEvolution(\mathcal{H})$, it is easy to see that $t_{k+1} > t_k$ for all $k \in \mathbb{N}$. Hence $CycleFree[g]$ is a sampling.

Assume that $CycleFree[g]$ is Zeno for some Q, t, x. Denote $t^* = sup_{k\in\mathbb{N}} t_k$ and $x^* = \Phi(t^*)$. If x^* is of local dimension $> (d'+2)^+$ the lemma is proved. Assume now that the the local dimension of x^* is $\leq (d'+2)^+$.

For all $k \in \mathbb{N}$, $HyperJump[g]$ must be Zeno for Q, t_{k-1}, x_{k-1}. As a consequence, by lemma 5.11 and by lemma 5.4, all the $x_k, k \in \mathbb{N}$ must be of local dimension $\geq (d'+1)$. By corollary 5.1, only a finite number of the $x_k, k \in \mathbb{N}$ must be of local dimension $\geq (d'+2)$, and only a a finite number of the $x_k, k \in \mathbb{N}$ must be of local dimension $(d'+1)$. Hence, there must exists some $k_0 \in \mathbb{N}$ such that for all $k \geq k_0$ x_k is of local dimension $(d'+1)^+$.

Apply lemma 5.4 on the subsequence $(x_k)_{k\geq k_0}$: There musts exists $k_0 \leq i_1 < i_2 \in \mathbb{N}, x^* \in \mathbb{Q}^d$ an a rational polyhedron F such that $Cycle(x_{i_1}, x_{i_2}, \mathcal{H}, Q, x^*)$ is true and such that the trajectory does not leave F between time t_{i_1}, t_{i_2}. Take i_1 and i_2 as the least integers such that the previous property hold and such that $i_2 - i_1 < i_1$. By definition 5.4 we have $(t_{i_2+1}, x_{i_2+1}) = Cycle_*((t_{i_1}, x_{i_1}), (t_{i_2}, x_{i_2}), \mathcal{H}, F, x^*)$. This is impossible since by lemma 5.4 this would imply that the local dimension of x_{i_2+1} is $\geq (d'+2)$.

□

5.4.3 Outputting recursive sampling

Lemma 5.13 *For all $k \geq 0$,*

- *one can construct a sampling $g_k : \mathbb{N} \times \mathcal{P} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d$ up to local dimension $(3+2k)^+$.*
- *For all $t \in \mathbb{R}, x \in \mathbb{R}^d$, denote by $R_{g_k}(t,x)$ the relation associated to g_k corresponding to (t,x).*

  There exists some $n_k \in \mathbb{N}$, $z_k \in O, |z_k| = \omega^k$ if $k \geq 1$, $|z_k| = 0$ if $k = 0$, such that, for all $t \in \mathbb{R}, x \in \mathbb{R}^d$, $W_{n_k}^{H^{\lceil (t,x)\rceil}(z_k)} = R_{g_k}(t,x)$

Proof: We prove the assertion by induction over $k \in N$.

The case $k = 0$ is corollary 5.2.

Assume $k \geq 1$. Consider $g_k = CycleFree[g_{k-1}]$. By lemma 5.12 and by induction hypothesis g_k is a sampling up to local dimension $(3+2k)^+$. By induction hypothesis, n'_{k-1} is a $H^{\lceil (t,x)\rceil}(z_{k-1})$-recursively enumerable index of $R_{g_{k-1}}(t,x)$ for all t, x.

Let $n \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ be fixed. Assume we have $H(y)^{\lceil (t,x)\rceil}$-recursively enumerable index m of $HyperJump[g_{k-1}](n,Q,t,x)$, where $m \in \mathbb{N}, y \in O$. By lemma 5.6, there exists a recursive r that maps m to

$r(m)$ where $r(m)$ is an an $H(y +_0 z_{k-1} +_0 1)^{\lceil (t,x) \rceil}$-recursive index of $R_{g_{k-1}}(HyperJump[g_{k-1}](n,Q,t,x))$. By lemma 5.11, there exists a fixed first order formula F such that for all $n \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, $\lceil HyperJump[g_{k-1}](n+1,Q,t,x) \rceil$ is definable by formula F from relation $R_{g_{k-1}}(HyperJump[g_{k-1}](n,Q,t,x))$ and from some recursive relations. By lemma 5.2, there exists $y_F \in O, |y_F| < \omega$ and a recursive g that maps $r(m)$ to $g(r(m))$, where $g(r(m))$ is an $H^{R_{g_{k-1}}(HyperJump[g_{k-1}](n,Q,t,x))}(y_F)$-recursively enumerable index of $\lceil HyperJump[g_{k-1}](n+1,Q,t,x) \rceil$. By lemma 5.6, there exists a recursive r' that maps $g(r(m))$ to $r'(g(r(m)))$, where $r'(g(r(m)))$ is an $H(y +_0 z_{k-1} +_0 1 +_o y_F)^{\lceil (t,x) \rceil}$-recursively enumerable index of $HyperJump[g_{k-1}](n+1,Q,t,x)$.

Denote by $h : \mathbb{N} \to O$ the recursive mapping such that $r(0) = 1, r(n+1) = r(n) +_0 z_{k-1} +_0 1 +_o y_F$ for all $n \in \mathbb{N}$.

As a consequence, for all $n \in \mathbb{N}$, $R^{<n}_{HyperJump[g_{k-1}]}$ is semi-recognized by the machine with oracle $H^{\lceil (t,x) \rceil}(h(n-1))$ that on input $< n, Q, P >$, compute for $i = 1, \ldots, n-1$ an $H^{\lceil (t,x) \rceil}(h(i-1))$-recursively enumerable index m_i of $HyperJump[g_{k-1}]$ (i,Q,t,x) from the $H^{\lceil (t,x) \rceil}(h(i-2))$-recursively enumerable index m_{i-1} of $HyperJump[g_{k-1}](i-1,Q,t,x)$ by the formula $m_i = r'(g(r(m_{i-1})))$ and then simulate the machine with oracle of number m_{n-1}. This machine has a fixed number independent of t, x.

Let $n \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$ be fixed. Assume we have $H(y)^{\lceil (t,x) \rceil}$-recursively enumerable index m of $CycleFree[g_{k-1}](n,Q,t,x)$, where $m \in \mathbb{N}, y \in O$. By lemma 5.6, there exists a recursive r that maps m to $r(m)$ where $r(m)$ is an $H(y +_0 h(n-1) +_0 1)^{\lceil (t,x) \rceil}$-recursive index of $R^{<n}_{HyperJump[g_{k-1}]}$ $(CycleFree[g_{k-1}](n,Q,t,x))$. By lemma 5.12, there exists a fixed first order formula G such that for all $k \in \mathbb{N}, Q \in \mathcal{P}, t \in \mathbb{R}, x \in \mathbb{R}^d$, $\lceil CycleFree[g_{k-1}](n+1,Q,t,x) \rceil$ is definable by G from relation $R^{<n}_{HyperJump[g_{k-1}]}$ $(CycleFree[g_{k-1}](n,Q,t,x))$ and from some recursive relations. As before, by lemma 5.2, and by lemma 5.6, there exists some recursive g and r' that maps m to $r'(g(r(m)))$ an $H(y +_0 h(n-1) +_0 1 +_o y_G)^{\lceil (t,x) \rceil}$-recursively enumerable index of $CycleFree[g_{k-1}](n+1,Q,t,x)$, for some fixed $y_G \in O, |y_G| < \omega$.

Denote by $l : \mathbb{N} \to O$ the recursive mapping such that $l(0) = 1, l(n+1) = r(n) +_0 h(n-1) +_0 1 +_o y_G$ for all $n \in \mathbb{N}$. Take $z_k = 3.5^p$ where $p \in \mathbb{N}$ is the number of recursive function l.

$R_{CycleFree[g_{k-1}]}$ is semi-recognized by the machine with oracle $H^{\lceil (t,x) \rceil}(z_k)$ that on input $< n, Q, P >$, compute for $i = 0, 1, \ldots, n$ an $H^{\lceil (t,x) \rceil}(l(i))$-recursively enumerable index m_i of $CycleFree[g_{k-1}](i,Q,t,x)$ from the $H^{\lceil (t,x) \rceil}(l(i-1))$-recursively enumerable index m_{i-1} of $CycleFree[g_{k-1}](i-1,Q,t,x)$ $(m_i = r'(g(r(m_{i-1}))))$ and then transform $H^{\lceil (t,x) \rceil}(l(n))$-recursively enumerable index m_n of $CycleFree[g_{k-1}]$ (n,Q,t,x) into a $H^{\lceil (t,x) \rceil}(z_k)$ index m of $CycleFree[g_{k-1}](n,Q,t,x)$ using lemma 5.5, and then simulate the machine with oracle of number m. This machine has a fixed number n_k. independent of t, x.

One has $|z_k| = \omega^k$.

□

5.4.4 Conclusion

Proposition 5.3 *Let $k \geq 1$.*

- *If a language L is semi-recognized by a PCD system of dimension $2k+3$ in finite continuous time then $L \in \Sigma_{\omega^k}$.*
- *If a language L is semi-recognized by a PCD system of dimension $2k+4$ in finite continuous time then $L \in \Sigma_{\omega^k+1}$.*

Proof: It is clear that for all $x \in \mathbb{Q}^d$, $\lceil x \rceil$ is recursive with a recursive index computable from x.

Let $k \geq 1$. By lemma 5.13, one can build a sampling g_k up to local dimension $(3+2k)^+$ and there exists some fixed n_k, and some fixed $z_k, z_k \in O, |z_k| = \omega^k$ such that, for all $t \in \mathbb{R}, x \in \mathbb{R}^d$, $W_{n_k}^{H^{\lceil (t,x) \rceil}(z_k)} = R_{g_k}(t,x)$.

Let $\mathcal{H} = (\mathbb{R}^d, f, \mathcal{J}, x^1, x^0)$ be a PCD system of dimension d recognizing language L.

Assume $d = 2k+3$: all the points of $\mathcal{H}$ have a local dimension $\leq (2k+3)$. As a consequence g_k can not be Zeno for any Q, t, x. L is semi-recognized by the machine with oracle $H(z_k)$ that on input $n \in \Sigma^*$, compute the $H(z_k)$-recursively enumerable index m of $S = R_{g_k}(0, \mathcal{J}(n), 0, \ldots, 0)$, and then by simulating $M_m^{H(z_k)}$, tests for $i = 0, 1, \ldots, \infty$ if $< i, x^1, x^1 > \in S$. It there is such an i, the machine accepts. If no i is found, the machine continues for ever.

Assume $d = 2k + 4$. By lemma 5.1, we know that all the points of local dimension d are rational points. Denote $Reach = \{< x, y, i > | x, y \in \mathbb{Q}^d \ i \in \mathbb{N} \ < i, y, y > \in R_{g_k}(0, x)\}$. $Reach$ is $H(z_k)$ recursively enumerable by the machine that on input $< x, y, i >$ computes the $H(z_k)$-recursively enumerable index m of $R_{g_k}(x)$, and then simulates $M_m^{H(z_k)}$ on input $< i, y, y >$. As a consequence, there exists a first order formula H with a quantifier $\exists$ and 0 alternation such that $Reach$ is definable from formula H from some $H(z_k)$-recursive relations: see [16].

Define the following relation $OneStep = \{< x, x^1 > | x \in \mathbb{Q}^d, x^1$ is a point of local dimension d and the trajectory starting from x reaches $x^1\} \subset \Sigma^*$. We claim that $OneStep$ is definable by some first order formula F from relation $Reach$: write F as the formula that says that x^1 is in $LocDim(\mathcal{H}, d' + 1)$ and that either there exists some $i \in \mathbb{N}$ such that $< x, x^1, i > \in Reach$, or g_k is Zeno for $x^1, 0, x$ and there exist some i_0 and some open polyhedron x' with $< x, x', i > \in Reach$, $x' \subset V_{x^1}$ and for all $i \geq i_0$, not $< x, V_{x^1}^c, i > \in Reach$, where $V_{x^1}^c$ is the complement of polyhedron V_{x^1} in $\mathbb{R}^d$.

If $< x, x^1 > \in OneStep$, it is clear that the formula must be true. Assume now that the formula is true: if there exists some $i \in \mathbb{N}$ such that $< x, x^1, i > \in Reach$, we are done: $< x, x^1 > \in OneStep$. Assume now that the second clause of the disjunction is true: we know that the trajectory starting from x is Zeno. Hence $(g_k(i, x^1, 0, x) = (t_i, x_i))_{i \in \mathbb{N}}$ is a converging sequence converging to some point x^* at time $t^* = sup_{i \in \mathbb{N}} t_i$. Since g_k is a sampling up to local dimension $(3+2k)^+$, x^* must be of local dimension d. Since Φ is Lipschitz, since V_{x^1} is an open polyhedron, we know that for some big enough i_1, for all $t_{i_1} \leq t < t^*$, $\Phi(t) \in V_{x^1}$. Hence, we must have $x^* \in \overline{V_{x^1}}$, where $\overline{V_{x^1}}$ is the topological closure of V_{x^1}. But, x^1 is the only point of local dimension d in V^{x^1}. Hence $x^* = x^1$.

See that formula F starts by a quantifier $\exists$ and has 1 alternation.

Now, see that L is definable by some first order formula G from relation $Reach$, from relation $OneStep$ and from some recursive relations: write that $n \in L$ iff there exists $m \in \mathbb{N}$, and an integer encoding m rational points $x_1, x_2, \ldots, x_m$, such that for all $1 \leq i < m < x_i, x_{i+1} > \in OneStep$, and $x_0 = (\mathcal{J}(\backslash), 0, \ldots, 0)$, and there exists some $i \in \mathbb{N}$, with $< x_m, x^1, i > \in Reach$.

Substitute every occurrence of relation $OneStep$ in formula G by formula F and every occurrence of formula Reach by formula H. One gets a resulting formula defining L from some $H(z_k)$ recursive relations starting with a quantifier $\exists$ and with 1 alternation. By lemma 5.2, $L \in \Sigma_2^{H(z_k)} = \Sigma_{\omega^k+1}$.

□

We get immediately from theorem 4.2 and from proposition 5.3:

Theorem 5.2 *Let $k' \geq 0$.*

- *The languages that are fully-recognized by a PCD system of dimension $2k'+3$ in finite continuous time are precisely the languages of $\Delta_{\omega^{k'}}$.*
- *The languages that are fully-recognized by a PCD system of dimension $2k'+4$ in finite continuous time are precisely the languages of $\Delta_{\omega^{k'}+1}$*
- *The languages that are semi-recognized by a PCD system of dimension $2k'+3$ in finite continuous time are precisely the languages of $\Sigma_{\omega^{k'}}$.*
- *The languages that are semi-recognized by a PCD system of dimension $2k'+4$ in finite continuous time are precisely the languages of $\Sigma_{\omega^{k'}+1}$*

Proof: The assertions for $k' = 0$ are immediate consequences of theorem 5.1 and of [6]. Now, assume $k \geq 1$. The two last assertions are proposition 5.3 and the two first assertions are immediate by considering the complement of the language recognized by the PCD system and by swapping the role of the point of acception and of the point of rejection.

□

In other words, we have a full characterization of the computational power of rational PCD systems.

References

[1] Eugene Asarin and Oded Maler. On some Relations between Dynamical Systems and Transition Systems. In *Proceedings of ICALP*, pages 59–72, 1994. Lecture Notes in Computer Science, 820.

[2] Eugene Asarin and Oded Maler. Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy. In *Proceedings of FSTTCS*, pages 471–483, 1995. Lecture Notes in Computer Science, 1026.

[3] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:33–65, 1995.

[4] Lenore Blum, Mike Shub, and Steve Smale. On a Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, July 1989.

[5] Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems. Proceeding of ICALP'97, July 1997.

[6] Olivier Bournez. Some bounds on the computational power of purely rational piecewise constant derivative systems. Technical report, LIP ENS-Lyon, 1997.

[7] Olivier Bournez and Michel Cosnard. On the computational power of hybrid and dynamical systems. *Theoretical Computer Science*, 168(2):417–459, 1996.

[8] Michael S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138:67–100, 1995.

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

[10] P. Koiran. Computing over the reals with addition and order. *Theoretical Computer Science*, 133:35–47, 1994.

[11] Pascal Koiran. A Weak Version of the Blum Shub Smale Model. Technical Report 005, NeuroCOLT Technical Report Series, August 1994. A preliminary version can be found in *Proceedings of 34th IEEE Symposium on Foundations of Computer Science*, pages 486-495, 1993.

[12] K. Meer and C. Michaux. A Survey on real Structural Complexity Theory. To be published in *Bulletin of the Belgian Mathematical Society - Simon Stevin.*

[13] Klaus Meer. A note on a $P \neq NP$ Result for a Restricted Class of Real Machines. *Journal of Complexity*, 8:451–453, 1992.

[14] Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162:23–44, 1996.

[15] P. Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and the foundations of mathematics.* Elsevier, 1992.

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