Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

A Geometrical Hierarchy on Graphs via Cellular Automata

Bruno Martin

Août 1999

Research Report N° RR1999-35

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

A Geometrical Hierarchy on Graphs via Cellular Automata

Bruno Martin

Août 1999

Abstract

Historically, cellular automata were defined on the lattices $\mathbb{Z}^n$, but the definition can be extended to bounded degree graphs. Given a notion of simulation between cellular automata defined on different structures (namely graphs of automata), we can deduce an order on graphs. In this paper, we link this order to graph properties and explicit the order for most of the common graphs.

Keywords: cellular automaton, graph of automata, simulation, Cayley graphs, graph growth, hyperbolic space, hierarchy.

Résumé

Historiquement, les automates cellulaires ont été défini sur les réseaux $\mathbb{Z}^n$, mais cette définition peut être étendue aux graphes de degré borné. Étant donnée une notion de simulation entre des automates cellulaires définis sur des structures différentes (c'est à dire entre des graphes d'automates), nous pouvons en déduire un ordre sur les graphes. Dans ce papier, nous relions cet ordre à des propriétés de ces graphes et explicitons l'ordre pour la plupart des graphes courrant.

Mots-clés: automate cellulaire, graphe d'automates, simulation, graphes de Cayley, croissance de graphes, espace hyperbolique, hierarchie.

A Geometrical Hierarchy on Graphs via Cellular Automata*

Bruno Martin

30th August 1999

Abstract

Historically, cellular automata were defined on the lattices $\mathbb{Z}^n$, but the definition can be extended to bounded degree graphs. Given a notion of simulation between cellular automata defined on different structures (namely graphs of automata), we can deduce an order on graphs. In this paper, we link this order to graph properties and explicit the order for most of the common graphs.

Introduction

Graphs of automata are extensions of cellular automata to connected bounded degree graphs. Their formalism was first introduced by P. Rosenstiehl under the name of "intelligent graphs" [14] and studied in [11]. Graphs of automata consist in connected bounded degree graphs with a coding and decoding neighbor vector and a state on each vertex. Each vertex state uniformly evolves, in discrete time, according to a definite rule involving the neighboring vertex states at previous time step ordered by the neighbor vector. The vertex states are updated simultaneously.

Simulations between cellular automata (CA) working on different architectures were studied in Róka [12, 13]. Other notions exist (for instance [7, 6, 2]), but we will focus on this definition which can be very naturally extended to graphs of automata. In fact this simulation definition is too permissive and allows counter-intuitive simulations. So we will add an additional hypothesis so that [12, 13] necessary condition of simulation would be true. This definition may be understood as a comparison between Cayley graphs, when they are considered as graphs of automata (whatever the involved finite automaton). Then, this notion of simulation between Cayley graphs may be extended to connected bounded degree graphs. We will here study the hierarchy induced by this simulation on the set of the connected bounded degree graphs.

In [12, 13], the notion of "simple" simulation between CA defined on Cayley graphs was linked to properties of groups. Here "simple" simulation denotes unitary simulation and means that each vertex state information is indivisible and that the simulation does not need complex operations like dispatching the information of a vertex state into an

*This report was submitted to a special issue of Fundamenta Informaticae

infinite number of vertices of the simulating graph and actually the definition of unitary simulation is not affected by the simulation definition correction. We will first exhibit a more powerful sufficient condition of "simple" (unitary) simulation by linking this notion to graph properties. But direct comparison between general graphs is not easy, so we will introduce an external space and study how the graphs can be embedded in this space. This method enables to compare most of the usual graphs and to specify the hierarchy.

In a first section, we will specify the notion of graph of automata and explain the simulations we will consider, the reason why we have to change the simulation definition and the new modified simulation definition will be explained in the second section. Some particular cases of simulation (elementary and unitary simulation), defined in the first section, will be studied in the third section: we will give a graph intrinsic characterization of elementary and unitary simulation. The fourth section describes a practical way to use this result. The consequences of these theorems on our knowledge of the hierarchy will be studied in the fifth section. In our last section, we will describe some open problems.

1 Definitions

We will first define graphs of automata (GA), that is cellular automata defined on connected bounded degree graphs. In addition to a set of states and a transition function, we need to order the neighbors of each vertex to know how to apply the transition function.

Definition 1.1 *A* graph of automata *(or GA) on a connected bounded degree graph $G = (V, E)$ is a 5-tuple (S, G, d, N, δ) where S is a finite set with at least two elements called* set of states*, G is a* graph *whose degree is bounded by d, $N : V \longrightarrow (\{1, ..., d\} \longrightarrow V)$ gives for each vertex its* neighbor vector*, i.e. an order on its neighbors and $\delta : (S \cup \{\epsilon\})^{d+1} \rightarrow S$ is the* transition function *where ϵ is a special element used when the vertex has less than d neighbors.*

A state is associated to each vertex of the graph. All the states change simultaneously in such a way that the next state in $v \in V$ is given by the neighbor vector of v: $N(v)$ and the local transition δ. If v has n neighbors ($n \leq d$) $N(v)(1), ..., N(v)(n)$, the next state s' of v is given by:

$$s' = \delta(state(v), state(N(v)(1)), state(N(v)(2)), ..., state(N(v)(n)), \epsilon, ..., \epsilon)$$

This local condition naturally defines a global transition on the automaton configurations. A configuration is a function $V \longrightarrow S$ which associates to each vertex its state. Let us define $C_{\mathcal{A}} = S^V$ the set of all configurations of the automaton $\mathcal{A}$. The GA transforms a configuration c into its successor $\mathcal{A}(c)$ such that:

$$\forall c \in C_{\mathcal{A}}, \forall v \in V, \mathcal{A}(c)(v) = \delta(c(v), c(N(v)(1)), c(N(v)(2)), ..., c(N(v)(n)), \epsilon, ..., \epsilon)$$

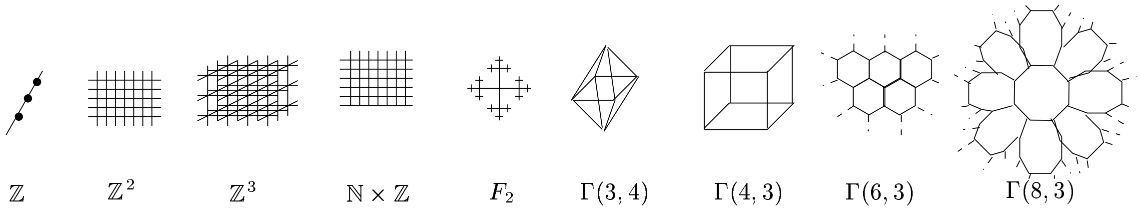

Figure 1: Some notations and examples of graphs

Notation 1.1 *Given a graph of automata $\mathcal{A}$, we will respectively denote $S_{\mathcal{A}}$, $G_{\mathcal{A}}$, $N_{\mathcal{A}}$ and $\delta_{\mathcal{A}}$ its set of states, its underlying graph, its neighbor vector and its local transition function.*

Notation 1.2 *Let us define the main graphs we will use (see figure 1)):*

- *the n-dimensional grid that we will denote $\mathbb{Z}^n$ to preserve Róka's notations (the n-dimensional grid is the Cayley graph of $\mathbb{Z}^n$);*
- *the half 2-dimensional grid, that we will denote $\mathbb{N} \times \mathbb{Z}$;*
- *the tree so that all its vertices have a degree equal to 4 which is the Cayley graph of F_2;*
- *the $\Gamma(k, d)$ $(k, d \geq 3)$ which are the graphs such as each vertex is surrounded by d k-gons (that is regular polygon with d edges).*

Now we recall the notion of simulation used in [12, 13]. A GA $\mathcal{A}$ is simulated by some GA $\mathcal{B}$ if we can encode the configurations of $\mathcal{A}$ into configurations of $\mathcal{B}$ such that after T steps of $\mathcal{B}$ we obtain the code of the next step of $\mathcal{A}$, formally:

Definition 1.2 (simulation definition without restriction on f)
Let $\mathcal{A}$ and $\mathcal{B}$ be two graphs of automata, $\mathcal{B}$ simulates $\mathcal{A}$ if there exists an injective function $f : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$ from the set of configurations of $\mathcal{A}$ into the set of configurations of $\mathcal{B}$ and a constant T such that for all the configurations $c \in C_{\mathcal{A}}$

$$f(\mathcal{A}(c)) = \mathcal{B}^T(f(c))$$

i.e. the following diagram commutes:

$$\begin{tikzcd} c \arrow[r, "f"] \arrow[d, Rightarrow, "\mathcal{A}"'] & f(c) \arrow[d, Rightarrow, "\mathcal{B}^T"] \\ \mathcal{A}(c) \arrow[r, "f"] & \mathcal{B}^T(f(c)) \end{tikzcd}$$

As we will prove in the next section, there is a mistake in [12, 13]: some conditions are implicitly assumed on f, but, actually, this definition allows some counter-intuitive and undesirable simulations. So in this paper, we will slightly modify it to the following one. The choice of this definition we be discussed in the following section. We start with the following notion of dependence:

Definition 1.3 *Let G_1 and G_2 be two connected bounded degree graphs. Let φ be a function from $Q_1^{G_1}$ into $Q_2^{G_2}$, we will say that the state of a vertex v of G_2 only depends on the states of a sub-set V_0 of G_1 vertices if and only if there exists a function $h : Q_1^{V_0} \longrightarrow Q_2$ such that for all configuration $c \in Q_1^{G_1}$, $\varphi(c)(v) = h(c|_{V_0})$ (i.e. the state of v in $\varphi(c)$ is given by applying h to the states of V_0).*

Example 1.1 *Let $\mathcal{A}$ be a GA on some graph G and v be a vertex of G, for each configuration $c \in C_{\mathcal{A}}$ of $\mathcal{A}$, the state of v in $\mathcal{A}(c)$ only depends on the states of v and its neighbors in c.*

Definition 1.4 (simulation definition with restriction on f)
Let $\mathcal{A}$ and $\mathcal{B}$ be two graphs of automata, $\mathcal{B}$ simulates $\mathcal{A}$ if there exists an injective function $f : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$ from the set of configurations of $\mathcal{A}$ into the set of configurations of $\mathcal{B}$ **such that there exists a constant $M > 0$ so that for each configuration c of $\mathcal{A}$, for each vertex v of $G_{\mathcal{B}}$, its state $f(c)(v)$ depends on less than M states of** *c, and there exists a constant T such that for all the configurations $c \in C_{\mathcal{A}}$*

$$f(\mathcal{A}(c)) = \mathcal{B}^T(f(c))$$

Notation 1.3 *The encoding function f of the simulation definition is assumed injective, so the co-restriction $f\,\big|^{f(C_{\mathcal{A}})}$ of f to $f(C_{\mathcal{A}})$ is bijective and we will for simplicity denote f^{-1} the function $\left(f\,\big|^{f(C_{\mathcal{A}})}\right)^{-1}$.*

Remark that there are few conditions over f, for instance f can burst the information of a state into different vertices (an unidimensional GA whose states are couples can be simulated by another unidimensional GA with the function

$$(\ldots,(s_0,s_1),(s_2,s_3),\ldots) \longmapsto (\ldots,(s_{-1},s_0),(s_1,s_2),\ldots))$$

In addition, some states may appear on an infinite number of vertices of the simulating graph (see the simulation of $\mathbb{Z}^2$ by F_2 in section 5). When this does not happen (i.e. when a vertex is simulated by only one vertex of the simulating graph), we will say that the simulation is *unitary*.

Definition 1.5 *A simulation is* unitary *if and only if*

$$\forall v \in G_{\mathcal{A}}\ \exists(!)v' \in G_{\mathcal{B}}\ \forall c \in C_{\mathcal{A}}\ \forall \alpha \in S_{\mathcal{A}}\ \exists \beta \in S_{\mathcal{B}}\ \ f(c_{v,\alpha}) = f(c)_{v',\beta}$$

where $c_{v,\alpha}$ is the configuration c in which the state of v is replaced by α.

Remark 1.1 *The notion of unitary simulation is the same with and without the condition over f because, as f is injective, if n states of $\mathcal{A}$ are stored in one state of $\mathcal{B}$, then the number $\#Q_{\mathcal{B}}$ of $\mathcal{B}$ states is at least $(\#Q_{\mathcal{A}})^n$. As the number of states of $\mathcal{B}$ is finite, only a bounded number of $G_{\mathcal{A}}$ vertices states may be stored in one $G_{\mathcal{B}}$ vertex.*

An unitary simulation defines an implicit function $G_{\mathcal{A}} \longrightarrow G_{\mathcal{B}}$ which associates to each vertex the only vertex of $G_{\mathcal{B}}$ which simulates it. If this function is injective, the simulation is said to be *elementary*.

From this definition, we deduce a pre-order on the connected bounded degree graphs in the following definition:

Definition 1.6 *Let G and G' be two bounded degree graphs, we will denote $G \leq G'$ and say that G' simulates G if and only if each graph of automata on G can be simulated by a graph of automata on G'.*

Remark 1.2 *Obviously, this relation defines a pre-order and thus an equivalence relation on the connected bounded degree graphs. We also obtain an order on the equivalence classes that we will call simulation order and study in the following sections.*

In a very natural way, we define the notions of *elementary* and *unitary simulation* when all the simulations can be elementary or unitary. We will prove in the third section that these two simulations defined on graphs through graphs of automata can be expressed intrinsically in graph theory.

2 Necessary Condition of Simulation

2.1 Zs. Róka's theorem

Let us recall here an important theorem of [12, 13]. We will prove in the next sub-section that this theorem is actually wrong with the definition 1.2 (without restriction on f), that is the reason why we modified the definition so that this theorem would be true for the definition 1.4 (see sub-section 2.3).

Definition 2.1 *Let G be a graph and v a vertex of G. Let us define the function $f_v : \mathbb{N} \longrightarrow \mathbb{N}$ which associates to n the number of vertices v' such that the distance between v and v' is less than n (i.e. such that there is a path between v and v' of length lower than n).*

Remark 2.1 *Let us notice that on a Cayley graph, the functions f_v are independent of the initial vertex v.*

We will admit the following lemma which shows that, on connected graphs, the asymptotic behavior of the functions f_v is independent of v. This allows us to define the graph growth of a connected graph G as the (common) asymptotic behavior of the functions f_v when v is a vertex of G.

Lemma 2.1 *Let G be a connected graph. If v, v' are two vertices of G, then $f_v = \Theta(f_{v'})$, that is, by definition, $f_v = O(f_{v'})$ and $f_{v'} = O(f_v)$ (i.e. there exists three positive constants M, λ and μ such that $\forall n > M \quad \lambda f_v(n) \leq f_{v'}(n) \leq \mu f_v(n)$).*

Definition 2.2 *Let G be a connected graph and v a vertex of G. The graph growth of G is the class of f_v among the set of functions $\mathbb{N} \longrightarrow \mathbb{N}$ quotiented by the relation Θ.*

By construction, the relation O induces an order on the set of functions $\mathbb{N} \longrightarrow \mathbb{N}$ quotiented by the relation Θ, but we will be interested in another relation to compare the graph growth which is defined below:

Definition 2.3 *Let α and β be two classes among the set of functions $\mathbb{N} \longrightarrow \mathbb{N}$ quotiented by the relation Θ. We will say that α is less than or equal to β and write $\alpha \leq \beta$ if and only if there exist some functions $f \in \alpha$ and $g \in \beta$ and some constants M, λ and μ such that $\forall n > M \quad f(n) \leq \lambda g(\mu n)$.*

Remark 2.2 *This definition is correct since it does not depend on the choice of the elements f and g and obviously define a pre-order. Let us notice that $3^n \neq O(2^n)$ and $3^n = 2^{\frac{ln(3)}{ln(2)}n}$ thus 2^n and 3^n are in two distinct classes, but are equal for the previously defined relation.*

Theorem 2.1 *Given two graphs G and G', if G' simulates G, then the graph growth of G is less than or equal to (for the definition 2.3) the graph growth of G'.*

The graph growth indicates the amount of information that a vertex can access to in n steps of the graph of automata evolution. This theorem just means that if whatever the slowdown constant T and even after a grouping operation (see below), a graph cannot access in nT steps to the information that another graph can access to in n steps, then the first graph cannot simulate the second one.

Let us specify here formally what we mean by a grouping operation. We can see a graph of automata as a graph with a finite automaton on each vertex. If it has enough states, a finite automaton can simulate a graph of automata defined on a finite graph. So given a graph of automata on G, and a finite subgraph G_0 of G, the behavior of the graph of automata on G_0 can be simulated by a finite automaton linked to all the neighbors in $G \setminus G_0$ of the vertices in G_0 without changing the behavior of the other vertices of G. We call this a grouping operation because the finite automata on each vertex of G_0 are grouped into one finite automaton with more states. By definition, a grouping operation leads to an unitary simulation. Of course, the composition of a finite number of grouping operations remains a grouping operation. In addition, we can perform simultaneously an infinite number of grouping operations if the number of grouped vertices is bounded, this leads to the following definition.

Definition 2.4 *Let G be a connected bounded degree graph and $(G_i)_{i\in I}$ a set of subgraphs of G whose number of elements is bounded by a constant M and such that, $\forall i \neq j,\ G_i \cap G_j = \emptyset$[1]. Let G' be the graph G where all the subgraphs G_i are successively replaced by vertices v_i linked to all the neighbors of the vertices of G_i in G. Let $\mathcal{A} = (S, G, d, N, \delta)$ be a graph of automata on G and $\mathcal{A}' = (S', G', d', N', \delta')$ a graph of automata on G'.*

Let us suppose that there exists $(f_i)_{i\in I}$ a family of injective functions $f_i : S^{G_i} \longrightarrow S'$ and an injection $\iota : S \longrightarrow S'$. $\mathcal{A}'$ is a grouped automaton of $\mathcal{A}$ if $\mathcal{A}$ is simulated without slowdown by $\mathcal{A}'$ with respect to the function

$$f \left| \begin{array}{l} C_{\mathcal{A}} \longrightarrow C'_{\mathcal{A}} \\ c \longmapsto c' \ \textit{where} \ \left\{ \begin{array}{l} \textit{if } v \notin (v_i)_{i\in I} \textit{ then } c'(v) = \iota(c(v)) \\ \textit{else } c'(v_i) = f_i(c|_{G_i}) \end{array} \right. \end{array} \right.$$

For instance, we can simulate a graph of automata on $\mathbb{Z}$ by another one on $\mathbb{Z}$ by grouping the states two by two. This simulation is done in real time, but let us notice that the simulating automaton has enough information to compute the two next steps. This kind of regular grouping over $\mathbb{Z}$ with acceleration is studied in [9]. But here we are not interested in any acceleration but in the possibility that each vertex of a simulating graph can simulate a bounded number of vertices of the simulated one.

[1] This implies that I is countable because G is countable (it is a connected bounded degree graph).

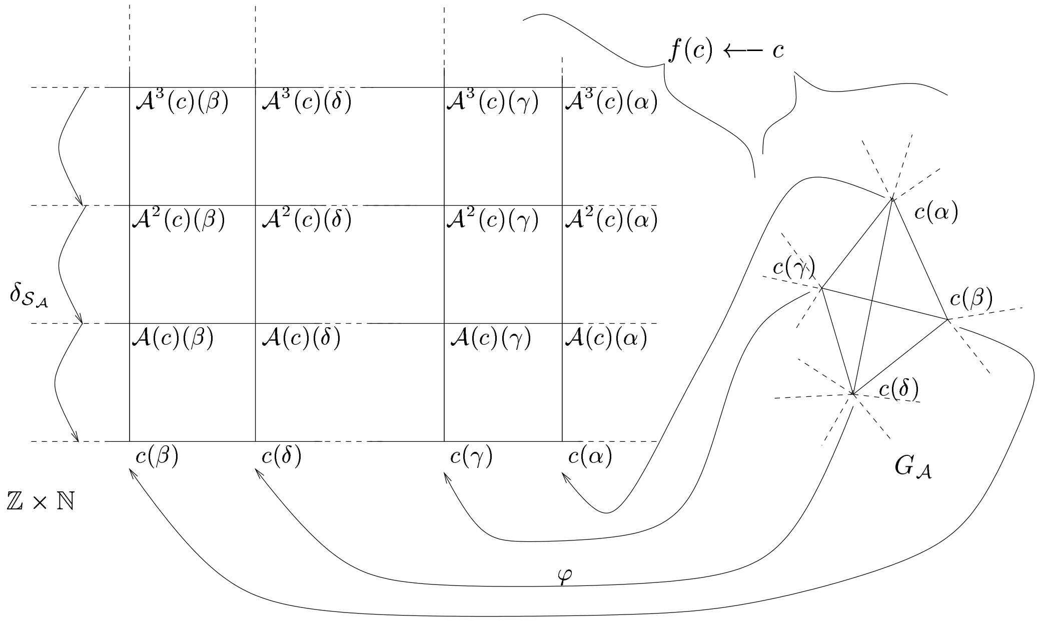

Figure 2: An undesirable simulation allowed if we add no restriction on f. Let c be a configuration of $\mathcal{A}$ and α a vertex of $G_{\mathcal{A}}$, $c(\alpha)$ represents the state of α in c. φ associates to α an half-line of $\mathbb{Z} \times \mathbb{N}$ on which we store the successive states of α starting from c: $c(\alpha)$, $\mathcal{A}(c)(\alpha)$, ...

After this specification about grouping operation, let us give two important corollaries of the theorem 2.1:

- $\mathbb{Z}^p \not\lesssim \mathbb{Z}^q$ if $p > q$ because $\mathbb{Z}^p$ growth behaves like n^p;
- $\forall p \;\; F_2 \not\lesssim \mathbb{Z}^p$ where F_2 is the free group with two generators because F_2 growth is $4 \times 3^{n-1}$.

2.2 All the infinite graphs are equivalents if we make no assumption on f

We will first prove a weaker result which proves that the theorem 2.1 is false without restriction over f and which is more intuitive and really easier to prove. Let us consider the graph $\mathbb{N} \times \mathbb{Z}$ i.e. an half 2-dimensional grid (see figure 2). Now, to each GA $\mathcal{A}$ on any bounded degree graph G, we design the GA $S_{\mathcal{A}}$ which has the same states as $\mathcal{A}$ and just perform a shift on each half line toward its origin (toward the bottom on the figure). It is easy to prove that $S_{\mathcal{A}}$ simulates $\mathcal{A}$ for the definition 1.2. Let us consider a bijection φ between G vertices set (which is countable) and the half lines of $G_{S_{\mathcal{A}}}$. To any configuration of $\mathcal{A}$, the function f assigns the configuration of $G_{S_{\mathcal{A}}}$ such that for a given vertex v of G, we store on $\varphi(v)$ at the position i the state of v after i computation steps.

Obviously, $G_{S_{\mathcal{A}}}$ graph growth behaves like n^2 and this graph simulates any connected bounded degree graph, that contradicts the theorem 2.1. So the definition 1.2 is not restrictive enough since we were able to encode the computation into the function f. This result raises a very natural question: is it possible to find, for each GA, a GA on a linear growth graph which simulates it? The answer is yes and we will prove a stronger result: there exists a GA defined on $\mathbb{Z}$ that simulates all the other GA (and even "extended GA" where the states number is just assumed at most countable). To prove this, we will use the evolution graph notion defined below:

Definition 2.5 *The evolution graph of a graph of automata $\mathcal{B}$ is the oriented graph $(\mathcal{C}_\mathcal{B}, A)$ where $C_\mathcal{B}$ is the set of $\mathcal{B}$ configurations and an arc (c, c') belongs to A if and only if $c' = \mathcal{B}(c)$, i.e. if and only if c' is the next configuration of $\mathcal{B}$ starting from the configuration c.*

Remark 2.3
- *By definition, every vertex of an evolution graph has an output degree equal to 1;*
- *an infinite bounded degree graph has a countable number of vertices, so the set of a GA configuration has at most the same cardinality than $\mathbb{R}$ which is $2^{\aleph_0}$[2], so a GA evolution graph has less than $2^{\aleph_0}$ vertices.*

Remark 2.4 *Let us consider the simulation definition 1.2, by definition a GA $\mathcal{B}$ simulates $\mathcal{A}$ without slowdown (i.e. $T = 1$) if and only if the $\mathcal{A}$ evolution graph is a subgraph of the $\mathcal{B}$ evolution graph (of course, it is wrong for the definition 1.4).*

Lemma 2.2 *There exists an oriented graph with $2^{\aleph_0}$ vertices and constant output equal to 1 that is universal in the following sense: all the oriented graph with less than $2^{\aleph_0}$ vertices and constant output equal to 1 are sub-graphs of this one.*

Proof 2.1 Let us consider a countable oriented graph with constant output degree equal to 1. We can decompose it into components, as the graph has less than $2^{\aleph_0}$ vertices, there are less than $2^{\aleph_0}$ components. If the graph is the evolution graph of $\mathcal{A}$ (we do not really need this condition for this proof, but we will use this notation because it is the case we are interested in), the components are the equivalence classes of vertices for the following relation:

$$cRc' \iff \exists \alpha, \beta \in \mathbb{N} \quad \mathcal{A}^\alpha(c) = \mathcal{A}^\beta(c')$$

The components may be of two distinct kinds: injective if it contains no cycle (i.e. it is a co-tree) or ultimately periodic of period P (see figure 3). If the graph is the evolution graph of $\mathcal{A}$, a component C_i is injective if and only if for all $c \in C_i$, and for all $n \in \mathbb{N}$, $\mathcal{A}^n(c) \neq c$. If not, there exists c and n such that $\mathcal{A}^n(c) = c$ and for all c' in the same component, $\exists \alpha, \beta \in \mathbb{N} \quad \mathcal{A}^\alpha(c) = \mathcal{A}^\beta(c')$ so the evolution from any of the component configuration leads to the same cyclic behavior.

If we consider a connected injective oriented graph G_I with a constant output degree equal to 1 and $2^{\aleph_0}$ antecedents at each vertex, this graph is in some kind universal among the connected injective oriented graphs with a constant output degree equal to 1 because all of them are obviously sub-tree of G_I. If we assign arbitrarily one vertex of a given injective oriented graph to one vertex of G_I, this implies the assignation of the

[2]We will admit here the choice axiom, so that $2^{\aleph_0}$ is a well defined cardinal. This is not absolutely necessary here, but it is simpler and, anyway, we will use the choice axiom in our proof.

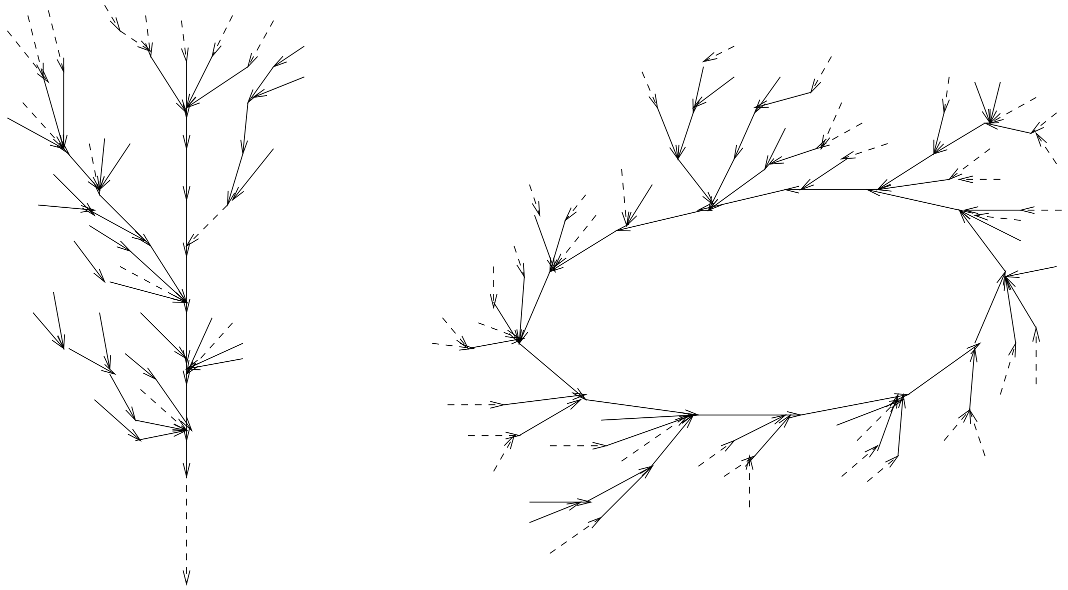

Figure 3: The components of an evolution graph may be of two kinds: injective as on the left diagram or ultimately periodic as on the right one

chain of its descendant (which correspond to its successor configurations). Then, when the graph is connected, by induction on the distance to one of the already assigned vertices, if the vertex v is assigned to v', the number of antecedents of v is at most $2^{\aleph_0}$ and the number of antecedents of v' is by hypothesis $2^{\aleph_0}$ so there exists an injection from the set of v antecedents into the set of v' antecedents[3]. A very similar proof show that a connected ultimately periodic (with period P) oriented graph G_P with a constant output degree equal to 1 and $2^{\aleph_0}$ antecedents at each vertex is universal in the same sense.

So a graph that is composed by $2^{\aleph_0}$ components like G_I and $2^{\aleph_0}$ components like G_P for all P is universal among the oriented graphs with less than $2^{\aleph_0}$ edges and constant output equal to 1. $\square$

Theorem 2.2 *All the infinite connected bounded degree graphs are equivalent for the definition 1.2 (if we make no additional assumption on f).*

Proof 2.2 We will actually prove that there exists a unidimensional GA which simulates without slowdown all the graph automata. This prove the expected result since the Cayley graph of $\mathbb{Z}$ is into the minimum class (see remark 1.2) among the infinite bounded degree graphs (this result will be proved for the simulation definition 1.4 (which is more restrictive) in the following sections).

We will define a unidimensional GA $\mathcal{U}$ (with 9 states) so that we can embed all the countable oriented graphs, with constant output degree equal to 1, into its evolution graph. So this GA simulates elementarily in time 1 all the other GAs.

[3]Let us notice that we implicitly use the choice axiom because we have to make simultaneously a possibly non countable infinite number of choices (of injections between the antecedents of v into those of v').

We will design $\mathcal{U}$ such that the universal graph designed in the proof of the lemma is sub-graph of $\mathcal{U}$ evolution graph. So, the $\mathcal{U}$ evolution graph must have $2^{\aleph_0}$ injective components and $2^{\aleph_0}$ ultimately periodic components of period P for each P. In addition, all these components will have a sub-graph with $2^{\aleph_0}$ antecedents at each vertex.

To design $\mathcal{U}$, we will arbitrarily separate the line into two parts: on the right, we will impose the ultimate behavior of the configuration (injective or ultimately periodic of a given period) and we will use the left part to ensure that the considered components of the $\mathcal{U}$ evolution graph have a sub-tree with $2^{\aleph_0}$ antecedents for each vertex (see figure 2.2).

- On the right, we just use one state l going toward the left (when it is surrounded by the background state bg), one state r going toward the right and two kinds of walls (sw and fw) on which the previous signal "rebounds" "slowly" on sw and "fastly" on fw. By that way, we can produce periodic behaviors of any period. Obviously, we have $2^{\aleph_0}$ possibilities to get a periodical behavior of a given period. To get an injective behavior, we can put cycles of all periods, again we have $2^{\aleph_0}$ ways to do this.
- Now, on the left, we need to ensure that all the previously defined components of the $\mathcal{U}$ evolution graph have a sub-tree with $2^{\aleph_0}$ antecedents for each vertex. The idea is to have candle-wick signals w that are consuming and then disappear. We need an infinite number of possible candle-wicks that end their consumption at a given time. To be able to stop an infinite number of candle-wicks at each computation step, we will use an infinite number of signals l going toward the left which are generated by the collision of one signal r going toward the right with fixed signals f (which say the r signal to $f$ork) lunched at the coordinates -2^n (see figure 2.2). Two additional states $r + f$ and $r + w$ represent the superposition of a $r$ight signal with a $f$ork or a candle-$w$ick one so that candle-$w$ick signals can cross r ones and $f$ork signals remain alive after generating a l to annihilate together with the next l signal. By that way, there are always an infinite number of l signals such that never in the past or in the future, another l signal has or will appear into the same cell. The infinite possible wicks that can end their consumption (by crossing a l signal) at each time step ensure that the designed configurations have $2^{\aleph_0}$ antecedents of the same kind.

□

2.3 Theorem 2.1 is true for the definition 1.4

As we saw before, the problem of the definition 1.2 is that to compute one state, the function f may access all the states of the antecedent configuration, and so, the function f can pre-compute all the GA evolution. To avoid this, we will add the assumption that f works "locally", that is each state of the image configuration will only depend on a bounded number of states from the antecedent configuration (this dependence notion has been defined in the first section).

To use [12, 13] proof we also need to assume that there exists at least one vertex on the simulated graph whose state depends on a finite number of the simulating graph

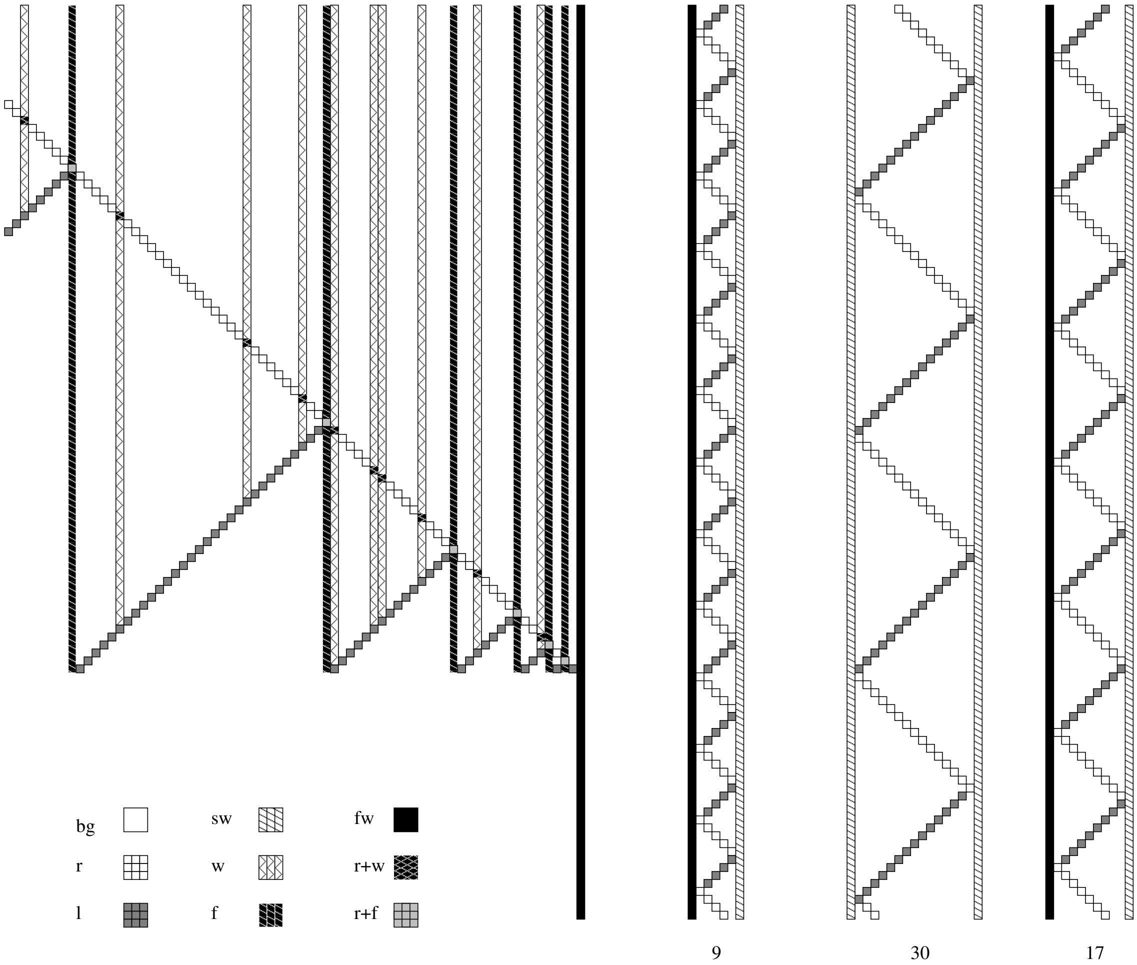

Figure 4: An example of $\mathcal{U}$ evolution (time goes toward the bottom), on the right, we see periodic behaviors of period 9, 30 and 17, and on the left the generation of *l*eft signals that stop the candle-*w*ick consumption

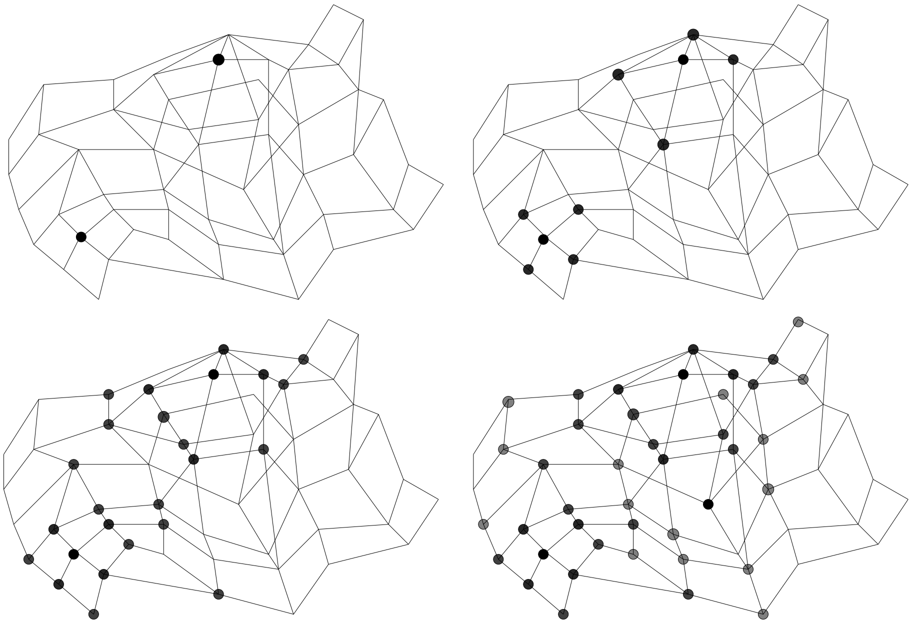

Figure 5: An example of ink blot GA

states. Let us prove that the theorem 2.1 is true under this two assumptions on f, we will see next that the second condition is actually a consequence of the first one.

Lemma 2.3 *Theorem 2.1 is true if we add the two following conditions on* $f : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$*:*

- *there exists* $M > 0$ *such that for each vertex* v *of* $G_{\mathcal{B}}$ *there exists a sub-set* V_0 *of* $G_{\mathcal{A}}$ *vertices with less than* M *elements such that* v *state only depends on the states of* V_0*;*
- *there exist a vertex* v_0 *of* $G_{\mathcal{A}}$ *such that its state only depends on the states of a finite sub-set of* $G_{\mathcal{B}}$ *vertices.*

Proof 2.3 With this additional condition, [12, 13] reasoning is true, it is a problem of information access. Let G be a graph of graph growth α and G' of graph growth β, with $\alpha > \beta$. Let us consider the expanding ink blot graph of automata on the graph G, that is the two states (0 and 1) graph of automata such that a cell remains in the state 0 if and only if none of its neighbors is in the state 1. So the 1 propagates in all directions at maximal speed, like an ink blot (see figure 2.3).

By refutation, we suppose that there exists a GA on G' which simulates the ink blot GA with a slowdown of T. Let us consider the vertex v_0 of G, by hypothesis, there exist v a vertex of G' and $\eta > 0$ such that the state of v only depends on vertices of G' whose distance to v is less than η, that is the vertices of $B(v, \eta)$. But after n steps of computation, the states of $B(v, \eta)$ only depend on the states of $B(v, \eta + Tn)$. Thus, by hypothesis, the states of $B(v, \eta)$ and so v_0 state only depend on $M \times \#(B(v, \eta + Tn)) = M \times f_v(\eta + Tn)$ states of the initial configuration of $\mathcal{A}$. As $\alpha > \beta$, there exists $n > \eta$

such that $\#(B(v_0,n)) = f_{v_0}(n) > M \times f_v((T+1)n) \geq M \times f_v(\eta + Tn)$. This is a contradiction since for all vertex $v \in B(v_0,n)$, its state after n steps of computation is not the same when we start from the "blank" configuration (all the states are 0) and when we start from the configuration where only the state of v' is one. □

We will now use some topological results. We will consider the product topology on the sets of configurations. For this topology, the elementary open sets are the sets of configurations so that the states of a finite number of vertices are fixed and the other one may takes any value. The support of an elementary set is the (finite) set of vertices on which the states are fixed. In addition, the sets of configurations are compact for this topology (Tychonoff) so we can use the following theorem which is classical in topology:

Theorem 2.3 *As Q^G is compact and f injective, the following conditions are equivalent:*

- *f is continuous;*
- *$f(Q^G)$ is compact and f^{-1} is continuous.*

Lemma 2.4 *Let φ be a continuous injective function from $Q_1^{G_1}$ into $Q_2^{G_2}$, for all vertex v of G_1, v state depends on a finite number of G_2 vertices states.*

Proof 2.4 Let $\mathcal{O}_q$ be the (elementary) open set of $Q_1^{G_1}$ elements such that v state is q. By construction,

$$\bigcup_{q \in Q_1} \mathcal{O}_q = Q_1^{G_1}$$

$$\text{so we have } \bigcup_{q \in Q_1} \varphi(\mathcal{O}_q) = \varphi(Q_1^{G_1})$$

and $\varphi(\mathcal{O}_q)$ are open sets of $\varphi(Q_1^{G_1})$ because f^{-1} is continuous. By definition, we can write the $\varphi(\mathcal{O}_q)$ as union of elementary open sets of $\varphi(Q_1^{G_1})$: $\varphi(\mathcal{O}_q) = \bigcup_{i \in \mathcal{I}_q} \mathcal{O}'_{q,i}$. As φ is continuous and $Q_1^{G_1}$ compact,

$$\varphi(Q_1^{G_1}) = \bigcup_{q \in Q_1, i \in \mathcal{I}_q} \mathcal{O}'_{q,i}$$

is compact, thus we can extract finite sub-sets $\mathcal{I}'_q$ of $\mathcal{I}_q$ such that

$$\varphi(Q_1^{G_1}) = \bigcup_{q \in Q_1, i \in \mathcal{I}'_q} \mathcal{O}'_{q,i}$$

Let us now consider the union V_0 of $\mathcal{O}'_{q,i}$ $(q \in Q_1, i \in \mathcal{I}'_q)$ supports and define the function $h : Q_2^{V_0} \longrightarrow Q_1$ such that if $e \in Q_2^{V_0}$ and $c \in Q_2^{G_2}$ is a configuration which coincide with e on V_0, we can define $q \in Q_1, i \in \mathcal{I}'_q$ such that $c \in \mathcal{O}'_{q,i}$, and $h(e) = q$. h is well defined because the open sets $\mathcal{O}'_{q,i}$ are elementary and so all the configuration that coincide on V_0 are in the same elementary open set by definition of V_0, thus the state of v depend on a finite number of states. □

Remark 2.5 *This lemma may be applied to f^{-1}, so f continuity ensure that the states of the simulating GA depend on a finite number of the simulated GA states, but this does not ensure that the number of states it depends on is bounded. Assuming that f is continuous is not sufficient to prove the theorem 2.1 since the simulations of any GA $\mathcal{A}$ by the shift GA $S_{\mathcal{A}}$ on $G_{S_{\mathcal{A}}}$ use a continuous encoding functions f.*

From the two following lemmas, we easily deduce the following result:

Theorem 2.4 *Theorem 2.1 is true for the simulation definition 1.4.*

3 Elementary and Unitary Simulations

Let $\mathcal{A}$ and $\mathcal{B}$ be two graphs of automata. If the simulation of $\mathcal{A}$ by $\mathcal{B}$ is unitary, then we can naturally define the function ϕ from $G_{\mathcal{A}}$ to $G_{\mathcal{B}}$ which associates to each vertex of $G_{\mathcal{A}}$ the only vertex of $G_{\mathcal{B}}$ which simulates it. If $\mathcal{A}$ is "complicated enough", each vertex needs the states of all its neighbors to compute its next state. But $\mathcal{B}$ simulates a step of $\mathcal{A}$ in T steps, so the distance in $G_{\mathcal{B}}$ between vertices simulating two neighbors of $G_{\mathcal{A}}$ is less than T. This exactly means that T is a Lipschitz constant for ϕ because connected graphs are (discrete) metric spaces, the distance between two vertices is the length of the smallest path between them and the Lipschitz condition definition is the following:

Definition 3.1 *Let $(\mathcal{E}, d_{\mathcal{E}})$ and $(\mathcal{F}, d_{\mathcal{F}})$ be two metric spaces. A function $f : \mathcal{E} \longrightarrow \mathcal{F}$ satisfies the Lipschitz condition if and only if*

$$\exists K \in \mathbb{R} \ \forall (x, y) \in \mathcal{E}^2 \quad d_{\mathcal{F}}(f(x), f(y)) \leq K d_{\mathcal{E}}(x, y)$$

We say that K is a Lipschitz constant for f.

In addition, only a finite number of vertices may be sent by ϕ on a given vertex because the simulating GA has only a finite number of states. So the existence of an injective (resp. with a bounded number of antecedents) function from $G_{\mathcal{A}}$ to $G_{\mathcal{B}}$ which satisfies the Lipschitz condition is necessary to have $G_{\mathcal{A}}$ simulated by $G_{\mathcal{B}}$ in an elementary (resp. unitary) way. In fact, this existence is also sufficient, as we will see next, so we have the two following results:

Theorem 3.1 *Let G and G' be two bounded degree graphs, G is elementarily simulated by G' if and only if there exists an injective function $f : G \longrightarrow G'$ which satisfies the Lipschitz condition.*

Theorem 3.2 *Let G and G' be two bounded degree graphs, G is unitarily simulated by G' if and only if there exists a function $f : G \longrightarrow G'$ with a bounded number of antecedents which satisfies the Lipschitz condition.*

To prove the sufficient parts of these theorems, the idea is to simulate each vertex by its image by the application. The images of two neighbors are separated by less than M vertices where M is the Lipschitz constant of f, so the graph of automata waits M steps to propagate the information and then the calculus is made thanks to a static information which indicates to each vertex the position of its neighbors in the simulated graph.

Proof 3.1 of both theorems 3.1 and 3.2

- Necessary part:
  Let us suppose that G' elementarily (resp. unitarily) simulates G and let us prove the existence of an injective (resp. with a bounded number of antecedents) function $f : G \longrightarrow G'$ which satisfies the Lipschitz condition. We consider the ink blot graph of automata on G with two states (0 and 1) that only propagates 1 to all its neighbors (see figure 2.3), this is the "complicated enough" GA we need. This GA is elementarily (resp. unitarily) simulated by a GA defined on G', let f be the function from G into G' which associates to each vertex v the only vertex of G' whose states depend on the state of v. f has a bounded number of antecedents because of the finite number of states of the simulating automaton (see remark 1.1). If the simulation is elementary, by definition, f is injective.

  Let T be the slowdown constant of the simulation, let us prove that T is a Lipschitz constant for f. Let x and y be two neighbor vertices of G. Let us consider the initial configuration where x contains the state 1, and all the other vertices contain the state 0. In T steps of the simulating GA, the state of $f(y)$ has been changed, so there is a path between $f(x)$ and $f(y)$ whose length is less than T.

- Sufficient part:
  For short we will prove that for injective functions f which satisfy the Lipschitz condition (theorem 3.1), there is an elementary simulation. If f has a bounded number of antecedents (theorem 3.2), the proof is exactly the same by a grouping operation (see definition 2.4), but of course, the simulation is no more elementary but unitary.

  Let f be an injective function satisfying the Lipschitz condition from G into G'. Let $\mathcal{A} = (S, G, d, N, \delta)$ be a graph of automata and N' be a neighbor vectors over G'. Let us build $\mathcal{B} = (S', G', d', N', \delta')$ on G' which elementarily simulates $\mathcal{A}$. With word, we will send the state of the vertex v of G into the state of the vertex $f(v)$. To simulate a step of $\mathcal{A}$, if M is a Lipschitz constant for f, M steps are enough to bring the states of the neighbors vertices. To calculate the next step of $\mathcal{A}$, the local transition function of $\mathcal{B}$ just needs to know which are the neighbors on G among all the vertices in the M radius ball on G'. The set of all possibilities is finite, so this information can be statically encoded in each vertex.
  Hence the finite set of states of $\mathcal{B}$ is:

  $$S' = ((S \cup \{\omega\}) \cup (S \cup \{\omega\})^d \cup ... \cup (S \cup \{\omega\})^{d^M}) \times (C \cup \{\epsilon\})$$

  where C represents the encoding of the neighbors in G (it is the finite set of functions $(S \cup \{\omega\})^{d^M} \rightarrow S^d$ which gives the neighbors in the simulated graph among the elements of the M radius ball in the simulating graph).
  Hence the function $g : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$ is defined as follows: if $v' \notin f(G)$ then the state of v' is (ω, ϵ). If $v' = f(v)$, v containing the state s, then its state will be (s, c), where $c \in C$ encodes the neighbors of v in G among the vertices in the M radius ball centered in v' in G'.

  δ' never changes the encoding part of the states. It stores during M steps the states of all its neighbors. Then, thanks to the encoding, it simulates the transition of $\mathcal{A}$: $\delta'((s, c)$, states of the neighbors vertices$) = (\delta(c(s)), c)$. So the slowdown constant is here $M+1$, but for simplicity the last step does not depend on the neighbors states. It is possible to do the last two steps in only one, the slowdown constant is in this case M.

□

Let us notice that there exist non unitary simulations, for example the simulation of $\mathbb{Z}^2$ by the free group with two generators F_2. We will prove in the fifth section that $\mathbb{Z}^2 \leq F_2$ (each vertex of $\mathbb{Z}^2$ will be simulated by an infinite number of vertices of F_2), and there is no unitary simulation because of the following theorem:

Theorem 3.3 *There exists no application satisfying the Lipschitz condition with a bounded number of antecedents from $\mathbb{Z}^2$ into F_2.*

Proof 3.2 Let us suppose by refutation that a function satisfying the Lipschitz condition f with a bounded number of antecedents exists. F_2 is a tree and as $\mathbb{Z}^2$ is not simulated by $\mathbb{Z}$ (see the second section for graph growth reasons), there are two branches of F_2 with an infinite number of vertices of $f(\mathbb{Z}^2)$.

So we can separate F_2 in two trees $F_2 = T_1 \cup T_2$ (see figure 6) with an infinite number of vertices in $f(\mathbb{Z}^2)$ in each one such that T_2 is a subtree of F_2 (so $T_1 = F_2 \setminus T_2$).

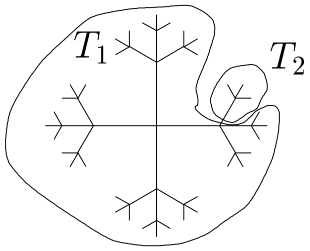

Figure 6: The partition of F_2, involved in the proof of the theorem 3.3

There are an infinite number of couples (x, y) of neighbor vertices in $\mathbb{Z}^2$ such that $f(x) \in T_1$ and $f(y) \in T_2$. To prove this, let us define $B_{f^{-1}(T_2)}$, the $f^{-1}(T_2)$ border (i.e. the set of all vertices of $f^{-1}(T_2)$ having a neighbor in $f^{-1}(T_1)$). Let us suppose by refutation that $B_{f^{-1}(T_2)}$ is finite, so it is included in a ball B of F_2. The vertices out of B are all either in $f^{-1}(T_1)$ or in $f^{-1}(T_2)$ because between each couple of vertices out of B, there is a path which does not cross the border. Hence $f^{-1}(T_1)$ or $f^{-1}(T_2)$ is included in B and so is finite which is impossible.

Now, as f satisfies the Lipschitz condition, all the elements of $f(F_{f^{-1}(T_2)})$ are at a bounded distance from T_1, which is again impossible because of the bounded number of antecedents. □

4 A Practical Way to Prove Unitary and Elementary Simulations

Given two graphs, it is not easy to know whether there exists an injective function from one graph into the other one which satisfies the Lipschitz condition. But let us try to

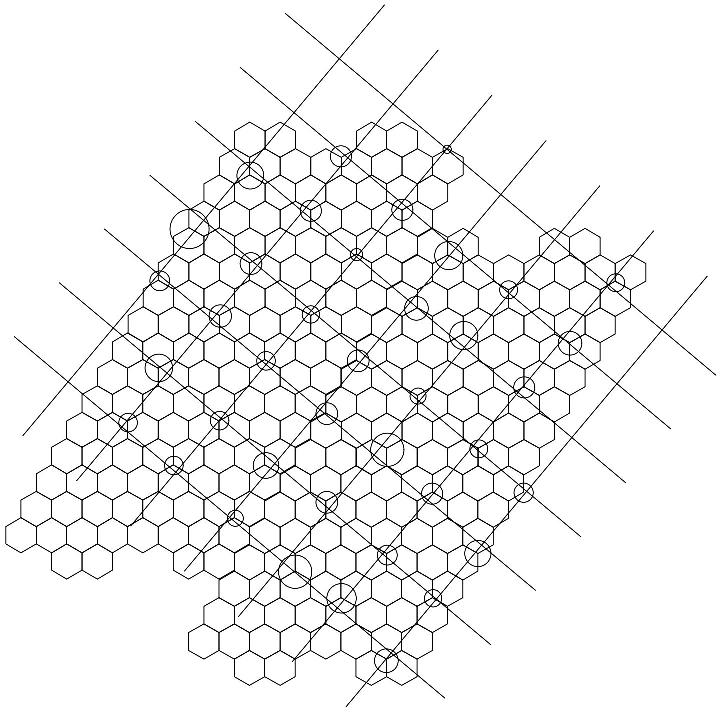

Figure 7: The grid $\mathbb{Z}^2$ can be simulated by the hexagonal tiling by simulating each vertex of the grid by the nearest vertex of the hexagonal tiling

find such an application from $\mathbb{Z}^2$, the two dimensional grid into the hexagonal tiling of the plane (see figure 7). We can associate to each vertex of the grid, the nearest vertex of the hexagonal tiling (if there are several nearest points, we chose one arbitrarily[4]). Clearly this application satisfies the Lipschitz condition and only a finite number of vertices may have the same image. So the grid is unitarily simulated by the hexagonal tiling. This reasoning is true for any embedding of the grid and the hexagonal tiling into the plane. So by similarity, we can reduce the scale of the hexagonal tiling such that the nearest point of two different vertices of the grid are always different (like on the figure 7). So in fact, there exist elementary simulations of the grid by the hexagonal tiling.

In the same way, all the tilings of the plane $\mathbb{R}^2$ are equivalent by elementary simulation, and furthermore for all n, all the tilings of $\mathbb{R}^n$ are equivalent to each other. We denote C_n the class of all the graphs equivalent to the tilings of $\mathbb{R}^n$ (including the n-dimensional grid $\mathbb{Z}^n$). All these classes are different because $\mathbb{Z}^n \leq \mathbb{Z}^m$ iff $n \leq m$ for graph growth reasons. We will now specify and improve our method to find functions between graphs which satisfy the Lipschitz condition.

To find an injective application from the grid into the hexagonal tiling satisfying the Lipschitz condition, we use an external space: $\mathbb{R}^2$. To be able to define the nearest vertex (or one of the nearest, because when several vertices are the nearest, we shall chose arbitrarily one of them), the external space needs to be metric and with compact closed balls. This last condition makes sure that the distance from a point to a closed set exists and is reached by a point of the set, so, given a point, we can define the (finite) set of the nearest points and chose arbitrarily one among them.

[4]Let us notice that we do not need the choice axiom here since the connected bounded degree graphs are countable. Actually, given a vertex of a GA graph, we can define an enumeration of its vertices using the neighbors vector and say we choose the vertex of lower index.

In addition, the simulated and simulating graphs do not need to be tilings of the external space. In fact, the simulated graph needs only to be embedded in this space with bounded vertices and a minimum distance between two vertices (so that only a finite number of vertices may have the same nearest point in the simulating graph). Graphs satisfying these two properties will be called plungings:

Definition 4.1 *Let $\mathcal{E}$ be a metric space with compact closed balls and $G = (V, E)$ be a graph. G can be plunged into $\mathcal{E}$ if there is a function $f : V \to \mathcal{E}$ such that:*

- $\exists \epsilon \in \mathbb{R}^{*+}$ *such that* $\forall x, y \in V,\ d_{\mathcal{E}}(f(x), f(y)) > \epsilon$ *if $d_{\mathcal{E}}$ represents the distance of $\mathcal{E}$ (i.e. there is a minimal distance between two points of $f(V)$);*
- *f satisfies the Lipschitz condition, i.e. $\exists M \in \mathbb{R}^+$ such that for all $(x, y) \in E$, $d_{\mathcal{E}}(f(x), f(y)) \leq M$.*

Remark 4.1 *For simplicity, we will identify the vertices of the plunged graph with their image in $\mathcal{E}$ by f.*

As for the simulated graph, the simulating one must satisfy the plunging properties and in addition cover all the space (i.e. the distance from any point of the space to the embedding of the simulating graph is bounded), and the graph distance has to be equivalent to the one induced by the space distance by lifting the embedding (to ensure the application will satisfy the Lipschitz condition). Such graphs will be said covering of the space, formally,

Definition 4.2 *Let $\mathcal{E}$ be a metric space with compact closed balls and $G = (V, E)$ be a graph. G is a covering of $\mathcal{E}$ if G can be plunged into $\mathcal{E}$ and if it satisfies the two following conditions:*

- $\exists M' \in \mathbb{R}^+$ *such that for all* $x \in \mathcal{E}$, $\exists v \in V$ *with* $d_{\mathcal{E}}(x, f(v)) \leq M'$ *(i.e. the distance from a point of $\mathcal{E}$ to $f(V)$ is bounded);*
- *The distance on G induced by the distance on $\mathcal{E}$ is equivalent to the distance of the graph G, that is there exists a constant k such that for all couples of vertices x and y, there exists a path from x to y whose length is less than or equal to $kd_{\mathcal{E}}(x, y)$.*

Let us remark that all the graphs that we intuitively want to call tilings of the space satisfy the covering properties, for instance the $\mathbb{Z}^n$ grid is a covering of $\mathbb{R}^n$.

The definitions of plunging and covering have been designed to ensure the existence of a function satisfying the Lipschitz condition with a bounded number of antecedents, so we can express the following theorem:

Theorem 4.1 *Let $\mathcal{E}$ be a metric space with compact closed balls, if G is a plunging into $\mathcal{E}$ and G' is a covering of $\mathcal{E}$, then $G \leq G'$ with a unitary simulation.*

Proof 4.1 We embed G and G' in $\mathcal{E}$ such that the embeddings satisfy the plunging and covering properties. Let g be the function from G to G' which associates to each vertex of G the nearest vertex of G' (if there are several nearest points, we chose one arbitrarily). g is well defined because $\mathcal{E}$ has compact closed balls: the embedding of G' in $\mathcal{E}$ is a closed set so the minimal distance between any point and this set is reached.

As the embedding of G' is discrete, only a finite number of vertices may be at the smallest distance.

Given a point of G', the set of points of $\mathcal{E}$ which are nearer this point of G' than any other point of G' is (by compactness of closed balls) included into the compact ball of radius M' by definition of covering. As only a finite number of balls with radius $\epsilon/2$ may be put into this ball, only a finite number of vertices of G may be mapped into the same vertex of G' by g.

In addition, if l is a Lipschitz constant for the plunging of G into $\mathcal{E}$, the distance in $\mathcal{E}$ of two images by g of two neighbors in G is bounded by $l + 2M$. But the distance in G' is equivalent to the distance in $\mathcal{E}$ so the distance in G' of the image by g of two neighbors in G is bounded by $k(l + 2M)$, so g is an application from G into G' which satisfies the Lipschitz condition.

So by the last section theorem, G' unitarily simulates G. $\square$

If similarities are possible on $\mathcal{E}$, that means practically that $\mathcal{E}$ is a normed vectorial space with compact closed balls, which is equivalent to $\mathcal{E}$ is a normed vectorial space with finite dimension, i.e. if $\mathcal{E} = \mathbb{R}^n$, then the simulations are elementary. For instance, here are some metric spaces which are not vectorial: the connected bounded degree graphs, the sphere, the cylinder, the cone, the paraboloïd of revolution and the hyperbolic space H_2 (see below). Let us notice that all the simulations deduced from the cylinder or the cone may be proved thanks to $\mathbb{R}$ or $\mathbb{R}^2$ and can also be elementary because a graph that covers (resp. can be plunged into) a cylinder (resp. a cone) covers (resp. can be plunged into) $\mathbb{R}$ (resp. $\mathbb{R}^2$).

Corollary 4.1 *If G is a plunging into $\mathbb{R}^n$ and G' is a covering of $\mathbb{R}^n$ then the simulation of G by G' is elementary.*

5 The Hierarchy

We will now consider the order defined by the remark 1.2. Let us recall that from the simulation between graphs of automata, we defined a simulation between connected bounded degree graphs. This simulation is obviously a pre-order and we naturally introduced the quotient set composed by the equivalence classes for this relation. We will say that a class simulates another one if and only if one graph of the first class simulates one graph of the second one, which is equivalent to say that each graph of the first one simulates all the graphs of the second one. By construction, this simulation notion on the set of equivalence classes is an order that we will study in this section. For simplicity, we will say that a graph G simulates (resp. is simulated by) a class C when the class of G simulates (resp. is simulated by) the class C (i.e. it simulates (resp. is simulated by) all the graphs of the class).

5.1 Main Features of the Hierarchy

The application of the last section theorems to the vectorial spaces $\mathbb{R}^n$ allows us to define the C_n classes (containing $\mathbb{Z}^n$). In addition, the simulations we can deduce over $\mathbb{R}^n$ are elementary thanks to the similarity possibility.

Many graphs are in none of the C_n like F_2 the free group with 2 generators (which has an exponential growth), and also graphs which can be plunged into $\mathbb{R}^{n+1}$ but not into $\mathbb{R}^n$ and which does not cover $\mathbb{R}^{n+1}$. For instance the graph of the figure 8 can be plunged into $\mathbb{R}^3$, but not into $\mathbb{R}^2$ and does not cover $\mathbb{R}^3$. Each graph of automata defined on this graph can be simulated by a GA defined on $\mathbb{Z}^3$. In addition, this graph simulates C_1 and is not comparable with C_2 although they have the same graph growth. The same idea allows us to build a graph simulated by $\mathbb{Z}^m$, incomparable with $C_i, n < i < m$ and simulating $\mathbb{Z}^n$: we link growing balls of $\mathbb{Z}^m$ like on the previous figure and connect it to the graph $\mathbb{Z}^n$.

Sz. Róka proved in [12, 13] that all the graphs $F_i, i > 1$ (i.e. the free groups with more than two generators) were equivalent for the unitary simulation. We denote their class by F. The following theorem proves that all the graphs can be simulated by a bounded degree (by d) tree and so by all the F_i ($i > 1$) because it is easy to inject this tree into F_d. Hence the F class is the maximum element for the simulation order. As a corollary, it proves that for all n, $C_n \leq F$.

Theorem 5.1 *Let G be a connected bounded degree (by d) graph. A developed graph of G is a tree with bounded degree (by d) and it simulates G (the simulation is not necessarily unitary).*

A developed graph is defined as follows:

Definition 5.1 *Let O be a generic vertex of G, the developed graph from O, $\mathcal{D}_O(G) = (V', E')$, of G is defined as follows: V' is the set of all the finite paths starting at O without going back. There is an edge between a and b if and only if they are paths like $a = (s_1, s_2, ..., s_n)$ and $b = (s_1, s_2, ..., s_{n+1})$, i.e. one is an initial segment of the other with exactly one more vertex.*

Let us remark that the developed graph of $\mathbb{Z}^2$ (which do not depend on O) is F_2, and that more generally developed graphs of Cayley graphs are the free groups. So the simulation of $\mathbb{Z}^2$ by F_2 works exactly like in the following proof. Each vertex of F_2 has the same neighborhood than the simulated vertex of $\mathbb{Z}^2$ so each vertex is simulated by an infinite number of vertices of F_2.

Proof 5.1 Each developed graph is clearly a tree whose degree in $x = (s_1, ..., s_n)$ is equal to s_n degree in G because x is linked to the vertex $(s_1, ..., s_{n-1})$ and to each vertex $(s_1, ..., s_n, s)$ where s is a neighbor of s_n and $s \neq s_{n-1}$ because going back is forbidden.

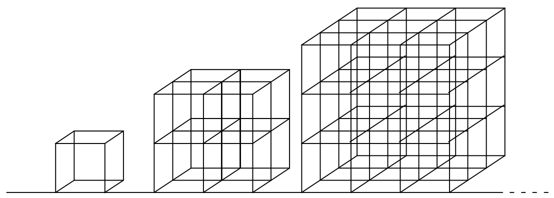

Figure 8: A graph with a square growth between C_1 and C_3 and not comparable with C_2

Let π be the canonical projection of $\mathcal{D}_O(G)$ into G (which associates to each path its last vertex). We will simulate the vertex v by all the vertices $\pi^{-1}(v)$ in $\mathcal{D}_O(G)$. This simulation is not necessarily unitary if, for some v, $\pi^{-1}(v)$ is infinite. The simulating automaton has exactly the same states and the same transition function. The neighborhood of each vertex in G is preserved by π so the evolution of the states in v and in each vertex of $\pi^{-1}(v)$ is the same. The simulation is done without lost of time. We can check that for each vertex v, if $v' \in \pi^{-1}(v)$, then v state depends on v' state, so each state depends on a bounded (by 1) number of the simulating graph vertices states. □

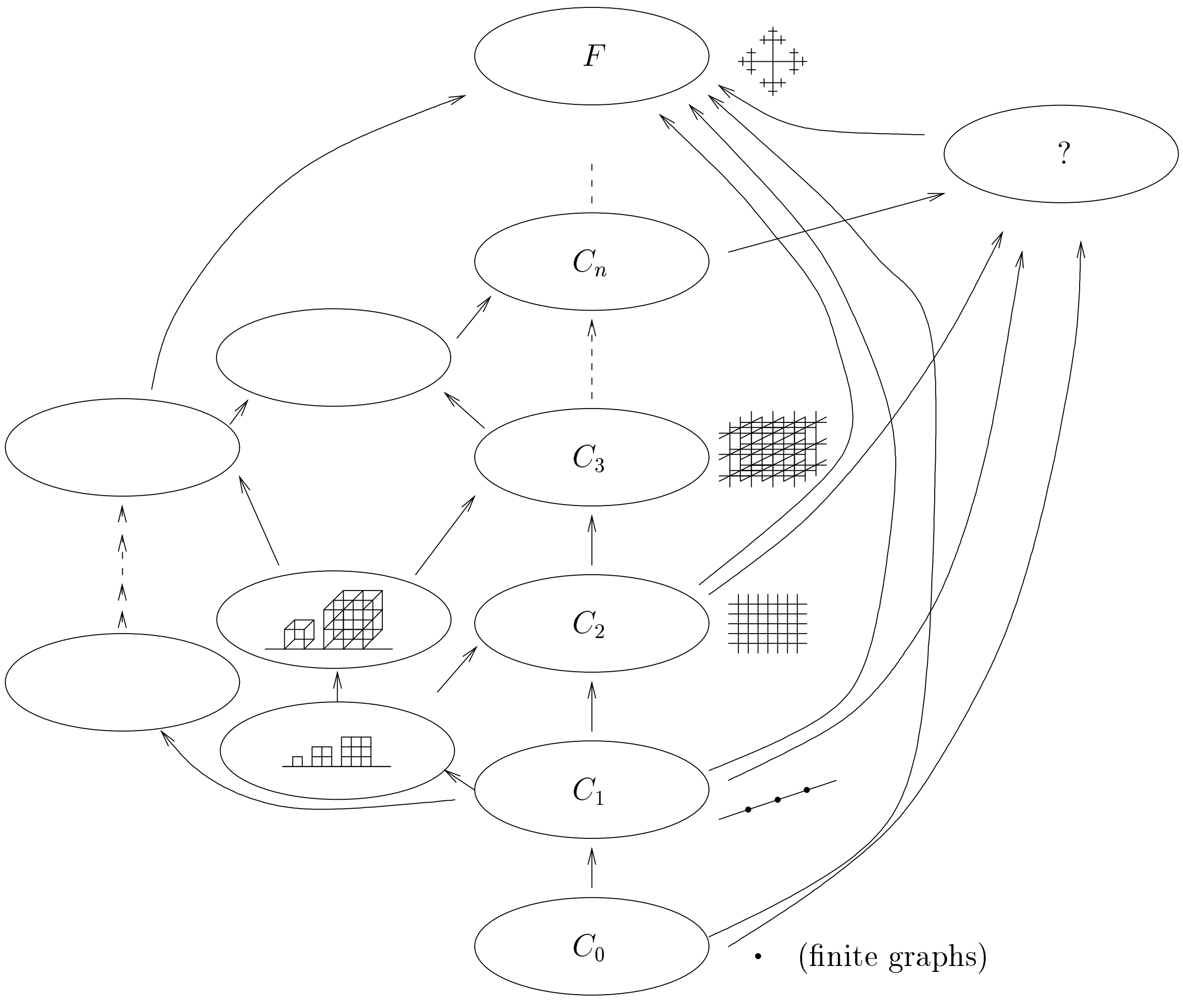

Figure 9: The graph hierarchy with relation to graphs of automata simulation

The hierarchy of graphs w.r.t. the simulation order can be represented as on the figure 9. Except C_0, the class of finite graphs, C_1 is smallest than all the other classes because they all contain a connected infinite graph and so an infinite path (and $\mathbb{N} \in C_1$). The class F is the maximum element of our order. All the other classes are greater than C_1, smaller than F and may be incomparable with some C_n. There exist classes simulating C_j, simulated by C_k and incomparable with the classes C_i, $j < i < k$ and classes simulating C_k and incomparable with the classes C_i, $i > k$. At last, between two classes C_i and C_j (except C_0 and C_1), there are an infinite number of classes.

5.2 Study of the Maximum Class

We will now study more precisely the class F. All the graphs in F have an exponential growth (else they could not simulate F_2) and so none can be plunged in any $\mathbb{R}^n$. Hence we will consider an other interesting metric space with compact closed balls: the 2-dimensional hyperbolic space H_2. Here is one of the definitions of H_2:

Definition 5.2 *H_2 is the unitary radius opened disk with the Riemann-metric defined by its scalar product at the point x by:*

$$g_x(v, w) = \frac{< v, w >}{(1 - ||x||^2)^2}$$

In this space, the perimeter of a circle with radius r is:

$$P = \frac{\pi(e^{2r} - 1)(e^{2r} + 1)}{2e^{2r}} \sim \frac{\pi}{2}e^{2r}$$

and so grows exponentially with the radius. So this space seems interesting to study F. A first application is to prove that F_2 can be plunged into H_2:

Theorem 5.2 *F_2 can be plunged into H_2.*

Proof 5.2 Let O be an arbitrary point of H_2. Let us consider all the points of F_2 whose distance to origin is n, we will put them regularly on the circle centered in O with perimeter $4 \times 3^{n-1}$ like on the figure 10. Let r_n be the radius of the circle of perimeter $4 \times 3^{n-1}$, $r_{n+1} - r_n$ is bounded (i.e. $\forall n \;\; m \leq r_{n+1} - r_n \leq M$). So the distance between two neighbors is less than $M + 1$ and more than m, this is a plunging into H_2. □

So all the tilings of H_2 are of course equivalent (for the unitary simulation) and are also in F. Let us now give some examples of tiling of H_2.

A regular polygon with k vertices is called a k-gon. Let $\Gamma(k, d)$ $(k, d \geq 3)$ be the graph such as each vertex is surrounded by d k-gons (see [1]). These graphs are studied with relation to the unitary simulation in [10]. For instance $\Gamma(4, 3)$ is the cube (which is in C_0), $\Gamma(6, 3)$ is the hexagonal tiling of the plane (and is in C_2). If $k = 3$ and $d \geq 7$, $k = 4$ and $d \geq 5$, $k = 5$ and $d \geq 4$, $k = 6$ and $d \geq 4$ or $k \geq 7$ then the graph cannot be embedded in any $\mathbb{R}^n$ because the angle sum in each vertex is greater than 2π.

In fact all these graphs are tilings of H_2. H_2 is the typical example of space with constant curvature -1. So the sum of a triangle angles is given by: $\hat{A} + \hat{B} + \hat{C} = \pi(1 - \mathcal{A}/\pi)$, where $\mathcal{A}$ is the triangle area, the sum is also between 0 and its value in Euclidean spaces: π. So for a k-gon, we have $\hat{A}_1 + ... + \hat{A}_k = (k-2)\pi(1 - A/(k-2)\pi)$. Hence, we can chose the scale of the k-gon such that its angles are $2\pi/d$, so we obtain a tiling.

Our theorem proves that all these $\Gamma(k, d)$ which cannot be embedded in any $\mathbb{R}^n$ are in F and equivalent for the unitary simulation.

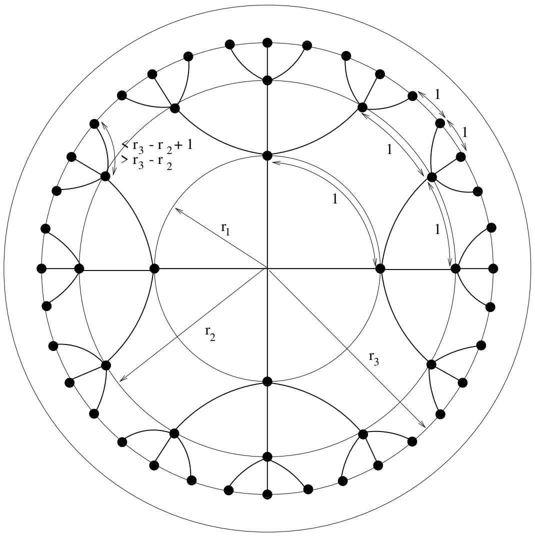

Figure 10: The plunging of F_2 into H_2

6 Discussion and Open Problems

The hierarchy of all the connected bounded degree graphs according to the intuitive notion of simulation defined in [12, 13] is not completely known, but we can classify most of the graphs because when we imagine a graph we naturally embed it in a known space.

In particular, we do not know whether a class different from F simulates all the classes C_n.

In addition, the link between graph growth and the hierarchy is enlightened because tilings of $\mathbb{R}^n$ spaces have an asymptotic growth in x^n and all the graphs in F have an exponential growth.

A graph cannot simulate another graph with a higher growth, but having a higher growth is not enough to simulate a graph (see figure 8). A natural question is to ask whether a "regular" graph can simulate all the graphs of smaller growth, and a first step would be to consider Cayley graphs (for a general introduction to Cayley graphs, see for instance [8]) and many questions are open:

Among finitely generated Cayley graphs (but with an infinite number of relations), some have neither polynomial, nor exponential growth (Grigorchuk [4], reformed and applied to language recognition by [3]), where are they in the hierarchy?

For finite presentation Cayley graphs, the existence of such graphs is still an open problem. So maybe all finitely presented Cayley graphs are in F or a C_n. This raises the two following questions:

- The underlying groups of Cayley graphs with polynomial growth are exactly virtually nilpotent groups [5]. These groups include commutative ones ($\mathbb{Z}^n$) and graphs like $U_3(\mathbb{Z})$ (the unipotent matrices like

$$\begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$$

where a, b, $c \in \mathbb{Z}$, see figure 11), which has 3 generators but its graph growth is equivalent to x^4. Is this graph equivalent to $\mathbb{Z}^4$?

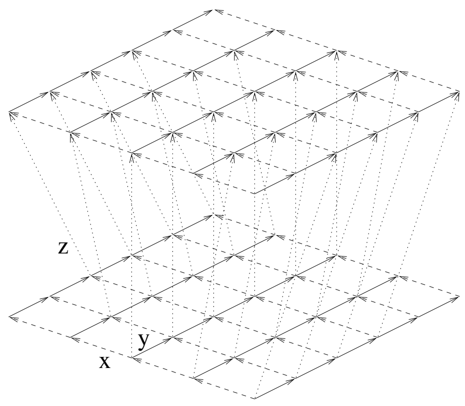

Figure 11: $U_3(\mathbb{Z})$ group: $< x, y, z \mid xy = yx, yz = zy, zx = xzy >$

- Are all exponential growth Cayley graphs in F?

A last interesting problem would be to find a more restrictive notion of simulation to classify more precisely the graphs of F.

Conclusion

The main structure of the hierarchy of the graphs with relation to the power of the graphs of automata that can be defined on them is now well known. Anyway, many problems remain open, the power of Cayley graphs like $U_3(\mathbb{Z})$ or Grigorchuk examples are still unknown.

Acknowledgment

The author whishes to thank Sylvie Ruette and Bill Allombert for numerous and fruitful discussions.

References

[1] T. Chaboud and C. Kenyon. Planar Cayley graphs with regular duals. *International Journal of Algebra and Computation*, 6(5):553–561, 1996.

[2] C. Choffrut and K. Čulik. On real time cellular automata and trellis automata. *Acta Informatica*, 21:393–407, 1984.

[3] Max Garzon and Yechezkel Zalcstein. The complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88(1):83–98, 30 September 1991.

[4] R. Grigorchuk. Degrees of growth of finitely generated groups and the theory of invariant means. *Math. USSR Izvestiya*, 25:259–300, 1985.

[5] M. Gromov. Groups of polynomial growth and expanding maps. *Publ. Math. I.H.E.S.*, 53:53–78, 1981.

[6] O. Ibarra, S. Kim, and S. Moran. Sequential machine characterizations of trellis and cellular automata and applications. *SIAM J. Computing*, 14:426–447, 1985.

[7] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. In *STACT'93*, volume 665 of *Lecture Notes in Computer Science*, pages 365–373, Würzburg, Germany, 25–27 February 1993. Springer-Verlag.

[8] R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer Verlag, 1977.

[9] J. Mazoyer and I. Rapaport. Inducing an order on cellular automata by a grouping operation. In *STACS'98*, volume 1373, pages 116–127. Lecture Notes in Computer Science, 1998.

[10] C. Nichitiu and E. Rémilla. Simulation of graph automata. Technical Report 19, Karlsruhe university, Fakultät für Informatik, Bibliothek - Postfach 6980 - 76128 Karlsruhe, 1998.

[11] E. Rémila. Recognition of graphs by automata. *Theoretical Computer Science*, 136:291–332, 1994.

[12] Zs. Róka. *Automates cellulaires sur graphes de Cayley*. PhD thesis, Ecole Normale Supérieure de Lyon, juillet 1994.

[13] Zs. Róka. Simulations between cellular automata on Cayley graphs. *Theoretical Computer Science*, to appear.

[14] P. Rosensthiel, J. R. Fiksel, and A. Holliger. Intelligent graphs: Networks of finite automata capable of solving graph problems. In R. C. Reed, editor, *Graph Theory and computing*, pages 210–265. Academic Press, 1973.