Computer Science Journal of Moldova, vol.**8**, no.2(23), 2000

A package of algorithms to devise cellular automata in the hyperbolic plane and related questions

Maurice Margenstern

Abstract

We introduce in this paper a new technique to deal with cellular automata in the hyperbolic plane. The subject was introduced in [7] which gave an important application of the new possibility opened by the first part of that paper. At the same time, we recall the results that were already obtained in previous papers.

Here we go further in these techniques that we opened, and we give new ones that should give better tools to develop the matter.

1 Introduction

Cellular automata have been studied for a long time, see [2, 3, 10], and they are most often used and studied in three spatial contexts: cellular automata displayed along a line, cellular automata in the plane, cellular automata in the three-dimensional space. There are also a few investigations in more general contexts, see for instance [13], where they are studied on graphs, connected with Cayley groups.

About the spatial representations, we should add the precision that in all cases, we are dealing with **euclidean** space. Indeed, that precision is so evident that it seems useless to remind this so *obvious* basis.

Take for instance cellular automata in the plane with von Neumann neighbourhood. If a cell has coordinates (x, y) with x and y integers, its neighbours are $(x, y+1)$, $(x, y-1)$, $(x-1, y)$ and $(x+1, y)$. This description is so simple that we forget the reason of so an elegant systems of coordinates, which extends without problem to the regular grids of the euclidean plane. Indeed, the group of displacements of the euclidean space possesses a normal subgroup, the group of translations and dilatations. That property namely is at the very basis of such elegant and simple coordinates.

The situation is completely different in the hyperbolic case, starting from two dimensions. The problem of finding an easy way to locate cells in that plane is not so trivial as it is in the euclidean case, because in the hyperbolic case, there are no equivalent to the euclidean group of translations and dilatations, because the group of hyperbolic displacements contain no nontrivial normal subgroup.

If the hyperbolic plane was considered in tiling problems, see [12], the study of cellular automata in that context was initiated by the technical report [6]. Later, two papers appeared, or will appear by the same authors, [7, 8] and two new technical reports by the present author are published, [4] and [5]. The present paper gives an account of [4] and [5], that are devoted to the representation of the regular rectangular pentagonal grid in the hyperbolic plane, that we call later the *pentagrid*. Here, we give new algorithms to locate cells of a cellular automaton grounded on the pentagrid. These new algorithms are simpler than the algorithm provided in [6]. Indeed they are linear in the size of the data.

As the basic features of what is needed of hyperbolic geometry are given in [6, 8] and [4], we shall not remind them here but simply remind the beginning of the proof of the existence of the pentagrid that is given in the just quoted papers. This reminding is necessary in order to understand the new tools that we indicate here.

2 Representations of the pentagrid

We remind that we take the Poincaré's disk as a model of the hyperbolic plane. The pentagrid is the tiling defined by the tessellation generated in the hyperbolic space by the regular pentagon with right angles. It is also called the regular rectangular pentagrid and we shall most often say simply *pentagrid*. Figure 1, below, illustrates the aspect of that grid restricted to a quarter of the hyperbolic plane that we defined in [6] and again in [4] to be the south-western quarter. It should be noticed that the pentagrid is the simplest regular grid of the hyperbolic plane. The triangular equilateral grid and the square grid of the euclidean plane cannot be constructed here as they violate the law about the sum of angles in a triangle which is always less than π in the hyperbolic plane.

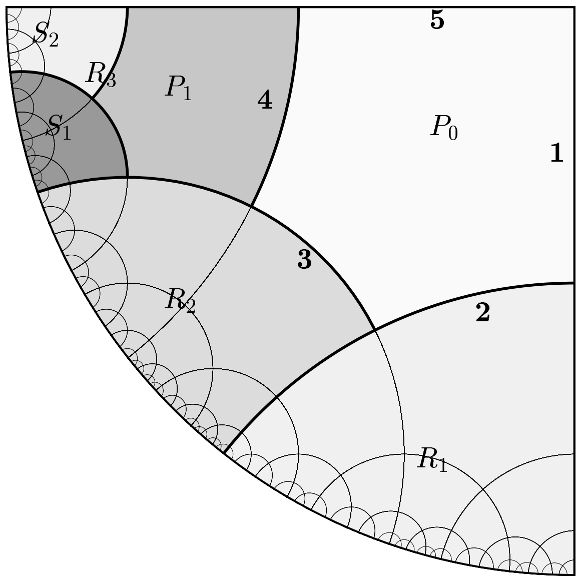
The existence and uniqueness of the pentagrid is a well-known fact of hyperbolic geometry. It was first proved by Henri Poincaré, [11], and other proofs were given later, for example in [1] and [9]. In [6], another proof is provided which gives rise to a feasible algorithm in order to locate cells. Here, we improved such an algorithm by constructing a new one, based on another principle. This gives rise to a family of algorithms, and we show that among them, there is a simplest one from the point of view of computer science.

2.1 Construction of the Fibonacci tree

The independent proof of the existence of the pentagrid is established in [6] by means of a bijection which is constructed between the tiling of the south-western quarter of the hyperbolic plane, say $\mathcal{Q}$, with a special infinite tree: the *Fibonacci tree.* Notice that $\mathcal{Q}$ is isometric to any quarter of the hyperbolic plane.

Let P_0 be the regular rectangular pentagon contained in $\mathcal{Q}$ that has one vertex on the center of the unit disk and two sides supported by the sides of $\mathcal{Q}$. Say that P_0 is the *leading* pentagon of $\mathcal{Q}$.

Number the sides of P_0 clockwise by **1**, **2**, **3**, **4** and **5** as indicated below, on figure 1. As **1** is perpendicular to **2** and **5** and as **4** is perpendicular to **3** and **5**, **2** and **3** do not intersect **5**. The complement of P_0 in $\mathcal{Q}$ can be split into three regions as follows. Line **2** splits $\mathcal{Q}$ into two components, say R_1 and R'_1 with R'_1 containing P_0. Line **3** splits R'_1 into R_2 and R'_2 with R'_2 containing P_0. Line **4** splits R'_2 into P_0 and R_3. This defines the initial part of a tree: P_0 is associated to the root of the tree, and let us consider that the root has three sons, ordered from left to right and respectively associated to R_3, R_2 and R_1. We can denote it as indicated by figure 1. We shall say that the root is a 3-node because it has three sons.



First step: regions P_0, R_1, R_2 and R_3, where region R_3 is constituted of regions P_1, S_1 and S_2;
Second step: regions R_1 and R_2 are split as the quarter (not represented) while region R_3 is plit into three parts: P_1, S_1 and S_2 as indicated in the figure.

Figure 1. Splitting the quarter into four parts

R_1 and R_2 are isometric images of $\mathcal{Q}$ by simple displacements: R_1 is obtained from $\mathcal{Q}$ by the displacement along **1** that transforms **5** into **2**. Similarly for R_2 with the displacement along **4** that transforms **5** into **3**. The same splitting into four parts can be repeated for these regions. Their leading pentagons are also 3-nodes.

Now, let us see the status of region R_3. It is plain that R_3 *is not* isometric to $\mathcal{Q}$. Let P_1 be the reflection of P_0 through **4** with sides which are now numbered anticlockwise, so that the same number is given to the edges supported by the same h-line. In order to avoid possible confusion, we put the name of the considered pentagon as an index, if needed. Say that P_1 is the leading pentagon of R_3. Notice that $R_3 \cup P_0$ is transformed into a region $\mathcal{S}$ by the displacement along $\mathbf{5}$ that transforms $\mathbf{1}_{P_0}$ into $\mathbf{4}_{P_0}$, say Δ, see figure 1. Define S_1 and S_2 as the respective images of R_2 and R_3 by Δ. Then notice that $\mathcal{S} = S_2 \cup P_1$. Say that S_1 and S_2 are the sons of R_3 and associate also these nodes to their leading pentagon. We say that the node associated to R_3 is a 2-node.

One can clearly see how me may proceed now. Define the following two rules:

- a 3-node has three sons: on left, a 2-node and, in the middle and on right, in both cases, 3-nodes;
- a 2-node has 2 sons: on left a 2-node, on right a 3-node.

Those two rules, combined with the axiom which tells that the root is a 3-node, uniquely define a tree which we call *Fibonacci* tree, see figure 2, below.

The properties of the Fibonacci tree are indicated in [6], [7] and [8], and they are thoroughly proved in [4]. We shall not remind all of them here, where our attention is focused on the *location* of the elements of the pentagrid. These properties are used in order to establish the following important result proved in [6], [7] and [8] that uses a cellular automaton based on the pentagrid.

Theorem 1 (Margenstern-Morita) NP-*problems can be solved in polynomial time in the space of cellular automata in the hyperbolic plane.*

2.2 A new tool, using the Fibonacci tree

Starting from [4], a new way is defined to locate the cells which lie in the quarter, by numbering the nodes of the tree with the help of the positive numbers. We attach 1 to the root and then, the following

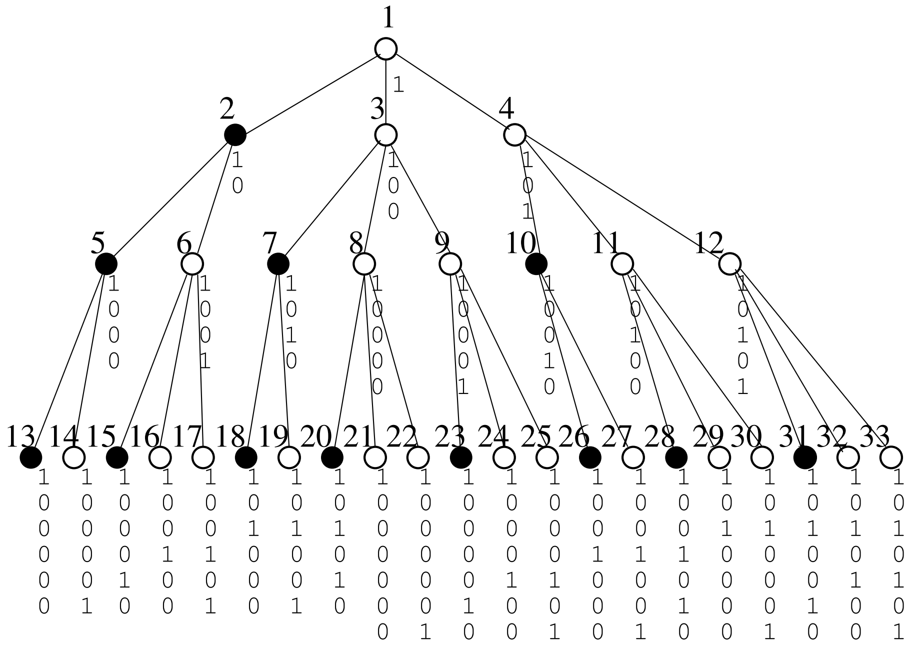


Figure 2. The standard Fibonacci tree: above a node: its number; below: its standard representation

numbers to its sons, going on on each level from left to right and one level after another one, see Figure 2, above.

That numbering is fixed once for ever in the paper. We fix also a representation of the numbers by means of the Fibonacci sequence, $\{F_i\}_{i\in I\!N}$.

It is known that every positive number n is a sum of distinct Fibonacci numbers: $n = \sum\limits_{i=1}^{k} \alpha_i.F_i$ with $\alpha_i \in \{0,1\}$. Such a representation defines a word $\alpha_k \ldots \alpha_1$ which is called a *Fibonacci representation* of n.

It is knwon that such a representation is not unique, but it can be made unique by adding a condition. Namely, we can assume that in the representation, there is no occurrence of the pattern 11: if $\alpha_i = 1$ in the above word, then $i = k$ or $\alpha_{i+1} = 0$. Following [4], we shall say that this new representation is the *standard* one. In [4], we give a proof of these well-known features.

From the standard representation, which can be easily computed from the number itself, it is possible to find the information that we need to locate the considered node in the tree: we can find its *status*, *i.e.* whether it is a 2-node or a 3-node; the number of its father; the path in the tree that leads from the root to that node; the numbers attached to its neighbours. This is done in great detail in [4] for the considered tree.

As we shall use another kind of Fibonacci trees, we shall not give more details about those tools that the interested reader may find in [4].

3 Constructing a continuum of Fibonacci trees

Now we show that there is indeed infinitely many ways to attach Fibonacci trees to the restriction of the pentagrid in a quarter of the hyperbolic plane.

3.1 A new Fibonacci tree

In order to see that, consider again figure 1. Indeed, that figure contains all the information that is needed in order to state the rules that lead to the tree represented in figure 2.

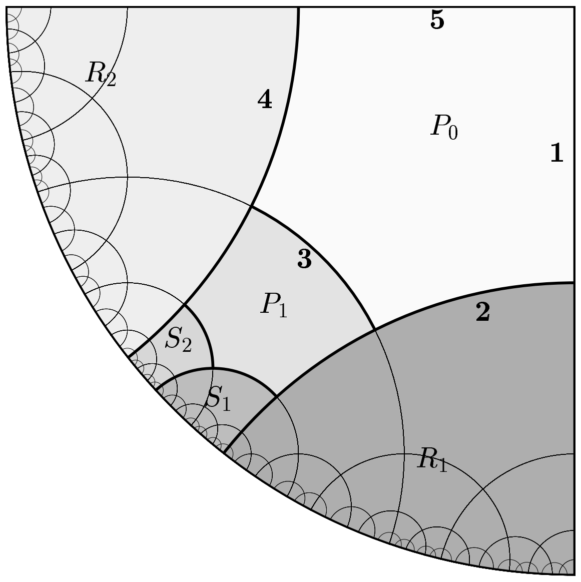
Indeed, we can split the quarter in another way, as shown by figure 3, below.

This defines a new splitting which differs from the one defined in [6, 7, 8] and [4], only on the way with which the regions that are isometric to a quarter are chosen.

At this point, we can notice that we can apply the arguments given in [6, 8, 4] in order to prove the bijection between the new tree and the tiling of the quarter. Indeed, when we consider diameter of regions that tend to zero as the index of the step of splitting tends to infinity, the estimated that we then established are still in force here.

Let us now focus on the trees that are obtained. The Fibonacci tree defined in the first papers can be rewritten as indicated in figure 2 where the numbers of the nodes are also displayed with their standard representation.

The new splitting that we define below gives rise to a new kind of Fibonacci tree, where the rules for the nodes are different for the 3-nodes. In the case of the Fibonacci tree, the rules can also be expressed as follows: $2 \to 2\,3$ and $3 \to 2\,3\,3$. In the case of this new tree, let us call it *central* Fibonacci tree, the rules are: $2 \to 2\,3$ and $3 \to 3\,2\,3$.



Region R_3 consists of P_1, S_1 and S_2.

Figure 3. Splitting the quarter into four parts in another way

As already indicated, the numbering of the nodes in the tree is fixed and so, the standard representation fixes the chosen Fibonacci representation. However, the algorithms which gives the status of a node, the number of the father, the path from the root to the considered node and the numbers of its neighbours will be different, see [5] for more details.

3.2 Infinitely many Fibonacci trees

It is now clear, that there are still other possible kinds of Fibonacci trees.

For a systematic study, one could proceed as follows. Using the previous notations, all possible rules for 2-nodes are $2 \rightarrow 2\,3$ and $2 \rightarrow 3\,2$ whereas all possible rules for 3-nodes are $3 \rightarrow 2\,3\,3$, $3 \rightarrow 3\,2\,3$ and $3 \rightarrow 3\,3\,2$. Denote these rules by, respectively, $2L$, $2R$, $3L$, $3M$ and $3R$. Also remind that the *status* of a 2-node is 2 and the status of a 3-node is 3.

Definition 1 *Call general Fibonacci tree, an infinite tree whose nodes have either two sons or three sons and such that there is a mapping τ of the nodes into the set $\{2L, 2R, 3L, 3M, 3R\}$ satisfying the following property:*

for all node ν, $\tau(\nu) \in \{2L, 2R\}$ if and only if ν is a 2-node.

Say then that τ matches the status of each node and that τ is a Fibonacci assignment, short an assignment, over the tree.

Due to the application to the tiling of a quarter of the hyperbolic plane by regular pentagons with right angles, we shall always assume that the root of the tree is a 3-node.

There are infinitely many general Fibonacci trees. They can be all constructed by a random algorithm using a dice[1] as follows:

- construct the root as a 3 node, which is at level 0;

[1] We use a *cubic*, hence euclidean, dice in a three-dimensional euclidean space

- iteratively construct levels one after another:

  for each node of the current level:

  - throw the dice and let r be the result;
  - for a 2-node apply rule $2L$ iff $r < 4$, otherwise $2R$;
  - for a 3-node apply rule $3L$ iff $r < 3$, else $3M$ if $r < 5$, otherwise $3R$.

As we have only permutations in the position of the 2-node among the sons of a node, this does not change the number of nodes which occur and, by induction on the level of the considered tree, it is easy to see that the number of nodes in a considered level is always the same for any general Fibonacci tree. Consequently, the numbering is always the same and, hence the standard representation attached to the numbers of the nodes only depends on the depth of the node in the tree, and on its rank on its level.

We can now state the following result:

Theorem 2 *There is a continuum of general Fibonacci trees and the trees are determined by their assignment.*

Among all possible assignments, we shall be also interested by the *fixed* ones: assign always the *same* 2-rule to 2-nodes and, similarly, always the same 3-rule to 3-nodes. There are six of them, in particular the Fibonacci tree, which corresponds to the assignment defined by rules $2L$ and $3L$, and the central Fibonacci tree, which is associated to the assignment defined by rules $2L$ and $3M$. From now on, call *standard* the assignment attached to the Fibonacci tree.

3.3 The preferred son property

In order to simplify the writing, we identify a node with its number and also with the standard representation of its number. We shall say that the representation of node ν ends in β, in short that ν ends in β, where $\beta \in \{0,1\}^*$, if β is a suffix of the standard representation of the number attached to node ν.

In [4], we noticed and proved the following property:

Proposition 1 *Let ν any node in the standard Fibonacci tree. Among the sons of ν there is exactly one, say ω, that ends in* 00*; moreover, the standard representation of ω is obtained by appending two* 0*'s to the standard representation of ν.*

We called *preferred son* the son with such a representation.

That property plays an important rôle in the algorithms that are given in [4]. It has also an important part in the algorithms that we introduce here, see [5].

The property of the preferred son is also true for the central Fibonacci tree. However, it is not true for any generalised Fibonacci tree. For instance, the tree built according to the rules $2R$ and $3R$ does not possess that property. Also the tree built according to the rules $2L$ and $3R$ does not satisfy that property. We refer the reader to [5] for more details.

Accordingly, following [5], we introduce the present definition:

Definition 2 *The continuator of the node numbered by μ is the node whose number ν is such that its standard representation is obtained by appending two* 0*'s to the standard representation of μ.*

When the continuator of a node happens to figure among its sons, the considered son will be called the *preferred son* of the node. If this happens for all the nodes of the tree, we shall say that the tree possesses the *preferred son* property.

In order to characterise the preferred son property, we first need to study the relations between any assignment with the standard one.

Relations between an assignment and the standard one

Let ν a node of level $k+1$ in the tree. We shall also identify ν with its *rank* among the nodes of the same level. Let α be a Fibonacci assignment. For each node different from the root, we associate a function f_α such that $f_\alpha(\nu)$ is the father, under that assignment, of the node with ν as a number. We also identify $f_\alpha(\nu)$ with its *rank* on its own level.

We shall denote the standard assignment by σ, and we shall see that in some sense, it possesses some maximal property.

Now, we define the following function on the nodes of a tree. Denote by $\omega_\alpha(\nu)$ the number of the rightmost son of ν under assignment α. A same node may have different sons under different assignments. To which extent can they be different? Not much as it is proved by the following relations, see [5] for the proofs:

Proposition 2 *For all k and ν we have:*

$$\begin{aligned} \omega_\alpha(\nu) - 1 \leq \quad & \omega_\sigma(\nu) \quad \leq \omega_\alpha(\nu) & (1)\\ f_\sigma(\nu) - 1 \leq \quad & f_\alpha(\nu) \quad \leq f_\sigma(\nu). & (2) \end{aligned}$$

Characterisation of the preferred son property

Call an assignment 00-*assignment* if and only if it possesses the following property: *under the assignment, each node has among its sons exactly one son whose number ends in* 00.

We have the following result:

Theorem 3 *A generalised Fibonacci tree possesses the preferred son property if and only if it is associated to a* 00*-assignment.*

This property is proved in [5] with theorem 1 and its corollary.

In [5], other characterisations of the preferred son property are proved. We indicate the following one:

Theorem 4 *An assignment α is a* 00*-assignment if and only if any node that ends in* 10 *is a son of a* 3*-node.*

In [5], we proved also the following result about the existence of trees with the preferred son property:

Theorem 5 *There is a continuum of generalised Fibonacci trees with the preferred son property.*

Fixed assignments

Among the assignments, some of them are good candidates for a convenient representation of the hyperbolic plane. In particular, *fixed* assignments are *a priori* to be first investigated.

However, as shown by the studies of [4, 5], fixed assignments do not have better properties than the standard one. Worse: as already noticed, two of them do not possess the preferred son property, see [5].

In connection with the 00-assignments, an ideal assignment would be a fixed 00-assignment such that the 2-son of a node is exactly its continuator. Unfortunately, there is no such fixed assignment. However, there is a 00-assignment such that the continuator of the nodes are exactly the 2-nodes. The assignment is *almost* a fixed one: the rule applied to the node depends on the ending of the node: 00, 01 or 10. See [5] for an exact construction with its proof.

4 Tools for the pentagrid

However, there is an assignment which is very near to what would be an ideal one.

4.1 The best assignment

We have seen that there is no fixed 00-assignment such that the 2-nodes would be exactly the continuators. What would happen if we would replace continuators by nodes that end in 01?

It can be seen that the rules $2R$ and $3M$ almost give the answer. Indeed, under that assignment, 2-nodes end in 01 except the nodes that are on the rightmost ranch or that are direct sons of nodes on that branch, see [5]. But now, a slight modification gives the answer:

Theorem 6 *There is a single* 00*-assigment such that* 2*-nodes are exactly the nodes ending in* 01*. The assignment consists in applying rule* $3R$ *to the root and then for all the other nodes, to apply rule* $2R$ *on the* 2*-nodes and rule* $3M$ *on the* 3*-nodes.*

From now on call 01-*assignment* the assignment constructed in theorem 6, which is illustrated on figure 4.

As it is proved in [5], the following properties hold:

- the sons of a 3-node end respectively in 00 for the leftmost one, in 01 for the middle one and in 10 for the rightmost one;
- the sons of a 2-node end respectively in 00 and 01;
- the continuator of a node is always its leftmost son.

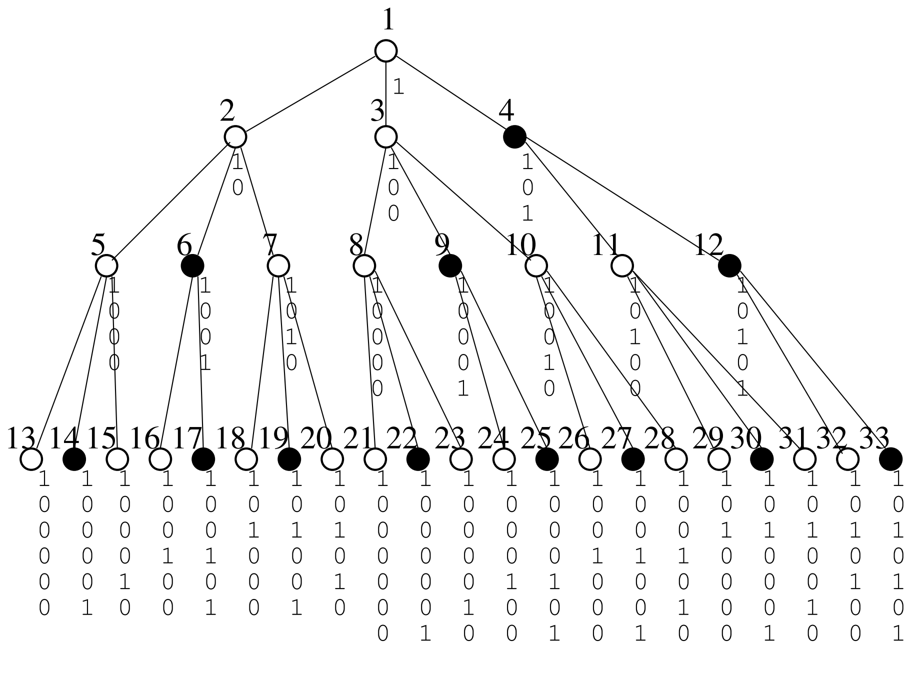


Figure 4. The Fibonacci tree associated to the 01-assignment

4.2 The algorithms

It is now clear that the 01-assigment is better fitted to our goals that any other one that we constructed before. Indeed, the rules to determine the status of a node are extremely simple and this simplifies also the rules for finding the path from the root to the node when we know the number attached to a node.

However, there is a small price to pay: the rules that give the reflections performed along a path are not exactly the same since the paths themselves are different from the paths of the standard Fibonacci tree. However, the new rules are not much more complex than the rules used in the standard case. We give them here, for positive orientation:

- if the node is a 2-node with i as the reflection leading to its father, then reflection i+2 leads to its left son which always end in 00, and reflection i+3 leads to its 2-son;
- if the node is a 3-node that ends in 00 with reflection i leading to its father, i+2 leads to its leftmost son, its continuator, i+3 leads to its middle son, the 2-son, and i+4 leads to the rightmost son, which ends in 10;
- if the node is a 3-node that ends in 10 with reflection i leading to its father, then i+1 leads to its leftmost son, its continuator, i+2 leads to its middle son, the 2-son, and i+3 leads to its rightmost son, which ends in 10.

In all cases, replace $+$ by $-$ if the orientation of the node is negative. The proof is analoguous to the one given in [4] for the standard situation.

For the completeness of this paper, now we indicate the algorithms needed for constructions using the pentagrid. Their correctness is straightforward from the proofs of theorem 6 and proceed by induction on the levels of the tree and on the rank for the level that follows the current one.

First of all, we start with the status of a node, the status being here with three values, detecting whether the node ends in 00, 01 or

10, giving respectively values 0, 1 and 2: take a to be the last digit of the stantdard representation of the node, b to be the penultimate digit, the status is then a+2.b.

Finding the father is also easy: erase the last two digits of the standard representation in order to have a representation of the father.

Algorithm to find the path

This algorithm needs an auxiliary one, which, to the path, associates the reflection through which the node is transformed into its father and conversely. This defines function *Index_Father* which is given by the first set of instructions that we give below in figure 5. Notice that in order to compute this function, the initialisation step proceeds outside the loop since the root is applied an exceptional rule.

With the help of functions *Index_Father* and *Father* as well as *Continuator* which appends two 0's to the standard representation of the node and then returns the corresponding number via *Value*, it is easy to find the neighbours of a node, as indicated on the right side of figure 5.

As indicated before, the proofs of the correctness of those algorithms rely mainly on the proof of theorem 6 which gives a lot of properties of the 01-assignment that are used in order to obtain the most straightforward computation as possible.

A complexity estimation

As we say that the provided agorithms are *simple*, such a statement must be proved by a complexity analysis:

Theorem 7 *The algorithms given in this section are linear in $log(n)$, where n is the number of the considered node, both in time and in space.*

The proof is straightforward from the examination of figure 5 where the algorithms are displayed. It is easy to notice that the algorithm that gives the standard reprsentation of n is linear in $log(n)$. The algorithm that gives the path from the number of the node is also linear in $log(n)$ as it is trivially linear in the length of the standard representation. As the function *Index_Father* is a loop on the path and as the corpus of the loop has a time complexity that is bounded by a constant, the linearity is clear. As the computation of *Father* and *Continuator* are

Algorithm for Index_Father

```
list is Path(node)
ref := 4 − path.top
case path.top is
  0 => status := 2
  1 => status := 0
  2 => status := 1
esac
pop path; sign is 1;
while path is not void
loop
  sign is − sign;
  if status is 0 or 1
    then ref :=
        ref+sign.(2+path.top);
    else ref :=
        ref+sign.(1+path.top);
  fi
  status := path.top
  pop path
pool
Index_Father is ref
```

Algorithm for the neighbours

```
status := Status(node)
fa := Father(node)
i := Index_Father(node)
s := Sign(node)
co := Continuator (node)
  neigh(i) := fa
case status is
  0 => neigh(i+s.2) := co
       neigh(i+s.3) := co+1
       neigh(i+s.4) := co+2
       if status(fa−1) = 1
       then neigh(i+s.1) :=
         fa−1
       else neigh(i+s.1) :=
         co−1
       fi
       neigh(i+s.1) := co−1
  1 => neigh(i+s.2) := co
       neigh(i+s.3) := co+1
       neigh(i+s.4) := co+2
       neigh(i+s.1) := co−1
  2 => neigh(i+s.1) := co
       neigh(i+s.2) := co+1
       neigh(i+s.3) := co+2
       neigh(i+s.4) := fa+1
esac
```

Notice that in the case of a node that ends in 00, the algorithm for the neighbours distinguishes between the two possible cases for the left brother of the father.

Figure 5. Algorithms to use the 01-assignment

also linear in $log(n)$, the computation of the neighbours has also a linear complexity.

For the whole plane

Notice that, for the sake of simplification, the latter algorithm uses a special kind of "+" operator. The addition is taken modulus 5 with also the convention that remainder 0 is written 5. The "−" operator, that is used when the orientation of the node is negative, is also understood in the same way, modulus 5 and with the same convention relative to 0. Details are left to the reader.

In order to represent the whole pentagrid with trees constructed with the 01-assignment, we proceed in the same way as it was done in [4] for the standard Fionacci tree. We remind the reader that we suggested in [4] to encode each quarter in a class of equal remainders modulo 4. In order to define the quarters, one of the diameters is called the vertical one, the other, the horizontal one. Then we may decide, as in [4], that multiples of 4 are devoted to the south-western quarter, that all remainders 1 go to the north-western one, that all remainders 2 go to the north-eastern one and that all remainders 3 go to the south-eastern one.

We have now to indicate how we may connect four trees associated to the 01-assignment.

The connection on the rightmost branch of the tree is made by 2-nodes. For such nodes, there is a missing connection when we consider the tree for the quarter. That connection can be used to connect, say, the south-western tree with the south-eastern one. The connection corresponding to $i{+}4$ (or $i{-}4$ if the orientation is negative) is made with the leftmost node of the south-eastern tree which is *on the same level*. This is possible: leftmost nodes on a branch are 3-nodes that end in 00 and applying the rule of the left brother of their father, we connect their $i{+}1$ (reps. $i{-}1$) arc with the rightmost node of the other tree again on the same level: this corresponds to the fact that the pentagons that lie along an extremal branch of a tree are reflected on the other quarter through the same reflection that defines that border. As the reflection is perpendicular to the line, an extremal node must be connected to a node of the same level in the other tree, see also [5] for a figure.

A remark

We already noticed that the endings 00 and 01 in the standard representation of the positive numbers occur quite often. We also already noticed that on another hand, the ending 10 is a bit less frequent. As we found 00-assignments for which the 2-nodes are exactly either all the nodes in 00, or all the nodes in 01, the question arises whether it is possible to find a 00-assignment with the same property for the nodes in 10?

The answer is no: assume that such an assignment exists. We know that each node contains its continuator among its sons. If a 2-node ends in 10, if its left son is its continuator, the right son cannot end in 10 and then no rule apply to this node. And this already happens for the leftmost node of level 1 which is 10. Its left son is its continuator and so the process cannot be continued.

5 Conclusion

And so, we have now at our disposal a lot of Fibonacci trees which all allow to locate cells of the pentagrid acurately. Our analysis proved that from the point of view of computer science, the better assignment is probably the 01-assignment.

This work is a direct continuation of [4] which opened a new way to locate cells on the pentagrid. As the quoted report, the method deals with a quarter of the hyperbolic plane.

A lots of questions remain open. We shall indicate two of them.

The first one concerns 00-assignments in general. We proved that there is a continuum of them. However, this does not indicate in some sense whether they are more numerous among all the assignments than the non 00-ones? As an example, we know that among the six fixed assignments, four ones are 00-assignments. Is it possible to say something more precise about such assignments than what was proved by theorem 5? Is there a probability for an assignment to be a 00-one and if the answer is yes, what is that probability?

Another question deals with with the other regular rectangular grids of the hyperbolic plane. It was indicated in [7] that the same construction based on the standard assignment can be generalised. Is it possible to say something more precise? At the time when this report is under printing, works are going on this second line by the author and Gencho Skordev, as announced in [4]. They contain a generalisation which also gives a new picture for the pentagrid. This will appear in a forthcoming preprint paper in the University of Bremen.

References

[1] C. Carathéodory. *Theory of functions of a complex variable*, vol.II, 177–184, Chelsea, New-York, 1954.

[2] M. Delorme and J. Mazoyer (eds.), *Cellular automata, a parallel model*, Kluwer Academic Publishers, **460**, 373pp., 1999.

[3] J. Gruska, *Foundations of computing*, International Thomson Computer Press, 716pp, 1997.

[4] Margenstern M., *Cellular automata in the hyperbolic plane*, Technical report, Publicationsdu GIFM, I.U.T. of Metz, N°99-103, ISBN 2-9511539-3-7, 34p. 1999.

[5] Margenstern M., *Cellular automata in the hyperbolic plane (II)*, Technical report, Publicationsdu GIFM, I.U.T. of Metz, N°2000-101, ISBN 2-9511539-.-., 40p. 2000.

[6] Margenstern M., Morita K., *NP problems are tractable in the space of cellular automata in the hyperbolic plane*. Technical report, Publications of the I.U.T. of Metz, 38p. 1998.

[7] Margenstern M., Morita K., *A Polynomial Solution for 3-SAT in the Space of Cellular Automata in the Hyperbolic Plane*, Journal of Universal Computations and Systems.

[8] Margenstern M., Morita K., *NP problems are tractable in the space of cellular automata in the hyperbolic plane*, to appear in Theoretical Computer Science.

[9] Maskit. B. *On Poincaré's theorem for fundamental polygons.* Advances in Math., **7**, 219–230, (1971).

[10] Morita K., *A simple construction method of a reversible finite automaton out of Fredkin gates, and its related model*, Transaction of the IEICE, **E**, 978–984, 1990.

[11] Poincaré H., *Théorie des groupes fuchsiens.* Acta Mathematica, **1**, 1–62, (1882).

[12] Robinson R.M. *Undecidable tiling problems in the hyperbolic plane.* Inventiones Mathematicae, **44**, 259-264, (1978).

[13] Zs. Róka, *One-way cellular automata on Cayley Graphs*, Theoretical Computer Science, **132**, 259–290, 1994.

Received July 6, 2000

Maurice Margenstern,
Groupe d'Inforamtique Fondamentale de Metz,
Laboratoire d'Informatique Théorique et Appilquée, EA 3097,
Université de Metz, I.U.T. de Metz, Département d'Informatique,
Île du Saulcy, 57045 Metz Cedex, France,
e-mail : margens@iut.univ-metz.fr