A weakly universal cellular automaton in the hyperbolic $3D$ space with three states

Maurice Margenstern

Automata 2010 — 16th Intl. Workshop on CA and DCS DMTCS proc. **AL**, 2010, 91–110

A weakly universal cellular automaton in the hyperbolic $3D$ space with three states

Maurice Margenstern

Université Paul Verlaine – Metz, IUT de Metz, LITA, Metz, France. Email: margens@univ-metz.fr

In this paper, we significantly improve a previous result by the same author showing the existence of a weakly universal cellular automaton with five states living in the hyperbolic $3D$-space. Here, we get such a cellular automaton with three states only.

Keywords: universality, cellular automata, hyperbolic geometry, $3D$ space, tilings

1 Introduction

In this paper, we follow the track of previous papers by the same author, with various collaborators or alone, see [2, 7, 13, 14, 10, 9], which make use of the same basic model, the *railway model*, see [16, 5, 9]. In order to be within the space constraint for the paper, we just refer to the above mentioned paper both for what is the railway model and for what is hyperbolic geometrY. For the latter one, we just mention something new in Section 2. A more developped version of the paper can be found on *arXiv*, see [11].

In the previous papers, the number of states of a weakly universal cellular automaton was reduced from 24 states to 9 ones in the pentagrid and fixed at 6 for the heptagrid. In [10], I succeded to reduce this number to 4 in the heptagrid.

The reduction for 6 states to 4 states, using the same model, was obtained by replacing the implementation of the tracks of the railway model. In all previous papers, the track is implemented as a one-dimensional structure where each cell of the track has two other neighbours on the track exactly, considering that the cell also belongs to its neighbourhood. The locomotive follows the track by successively replacing two contiguous cells of the track: the cells occupied by the front and by the rear of the locomotive. The locomotive has its own colours and the track has another one which is also different from the blank, the colour of the quiescent state. In the mentioned paper, this traditional implementation is replaced by a new one. There, the track is no materialized but suggested only. It is delimited by *milestones* which may not define a continuous structure.

At this point, my attention was drawn by a referee of a submission to a journal explaining the 4-state result that it is easy to implement rule 110 in the heptagrid, using three states only. This is true, but this trick produces an automaton which is not really a planar automaton and does not improve our knowledge neither on rule 110 nor on cellular automata in the hyperbolic plane. This implementation with three states can also be easily adapted to the dodecagrid of the hyperbolic $3D$ space and suffers the same defect of bringing in no new idea.

In this paper, we follow the same idea of milestones as in [10]. Here too, the milestones are implemented in two versions. However, thanks to the third dimension, the same pattern can be used to change directions, either inside a plane of the hyperbolic $3D$ space or to switch from one plane to another one. This configuration is used to avoid crossings, replacing them by bridges, as this was already performed in [7]. Sections 3 and 4 thoroughly describe the implementation of the model in the hyperbolic $3D$ space. Section 5 explains how to check the rotation invariance of the rules. For the correctness of the rules themselves, we refer the reader to [11] where they are fully listed. In Section 5, we also give a short account on the computer program which we used to perform the simulation and to check the correctness of the rules.

This will conclude our proof of the following result:

Theorem 1 (Margenstern) – *There is a cellular automaton in the dodecagrid of the hyperbolic* $3D$ *space which is weakly universal and which has* 3 *states. Moreover, the cellular automaton is rotation invariant and its motion actually makes use of the three dimensions.*

By the latter expression, we mean that the automaton cannot be reduced to a lower dimension by a simple projection. We refer the reader to [9, 7] for a discussion on the notion of weak universality. The reader is also referred to [11] for figures and tables, not included here in order to comply to page constraints.

2 Navigation in the dodecagrid

Here, we use the ideas of [8, 9] to define navigation tools for the dodecagrid. Using Schlegel diagrams, see [7, 8, 9], we can also define a splitting of the hyperbolic $3D$ space into 8 corners around a point which is a common vertex. Next, we split the corner as indicated in Figure 1. The idea of the representation

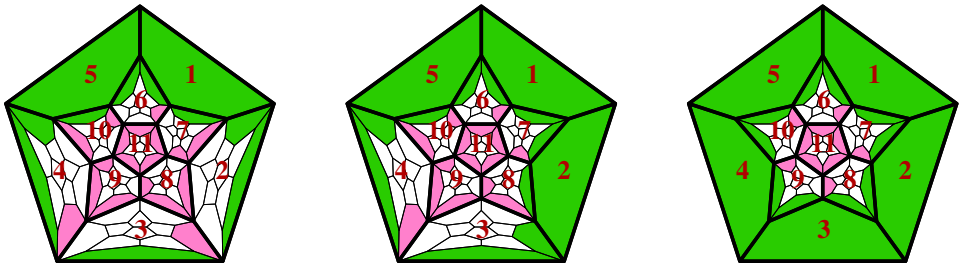


Fig. 1: Splitting a corner of the dodecagrid. On the left-hand side, the splitting of a corner. On the middle, the splitting of a half-octant. On the right-hand side, the splitting of a tunnel. Note that the faces are numbered according to the convention introduced in [7, 9].

is, as in the pentagrid, to fix rules which allow to get a bijection of a tree with the tiling restricted to the corner. If we allow the reflection of any dodecahedron of the tiling in its faces, we shall get many doubled replications as explained in [8].

We refer the reader to [8]. However, the splitting suggested by Figure 1 is a bit different from that indicated in [8]. The difference is that the splitting of Figure 1 is more symmetric and it involves three basic regions instead of four ones in the splitting of [8].

In the paper, we shall not directly use the tree. Taking into account that most of the circuitry will occur in a plane Π_0, we shall use projections onto Π_0. Now, we can chose Π_0 to be plane of a face of a fixed dodecahedron. In this way, the restriction of the dodecagrid to Π_0 is a copy of the pentagrid. And so, for the projections we have in mind, we can use the pentagrid.

In Π_0, each tile of the pentagrid is a face of exactly one tile of the dodecagrid over Π_0. We draw a Schlegel diagram of the corresponding dodecahedron within its face which lies on Π_0. We shall call this a **pseudo-projection onto** Π_0. Imagine that we have four tiles O, Y, G and B defined by their respective colour, orange, yellow, green and blue. Imagine that another tile W, a white one, sees Y through its face 1, the same face being numbered 5 in Y, the face 1 of Y being that which is shared with G. Then, we can see two other tiles on W, a red one and an orange one, on faces 10 and 6 respectively. On the left-hand

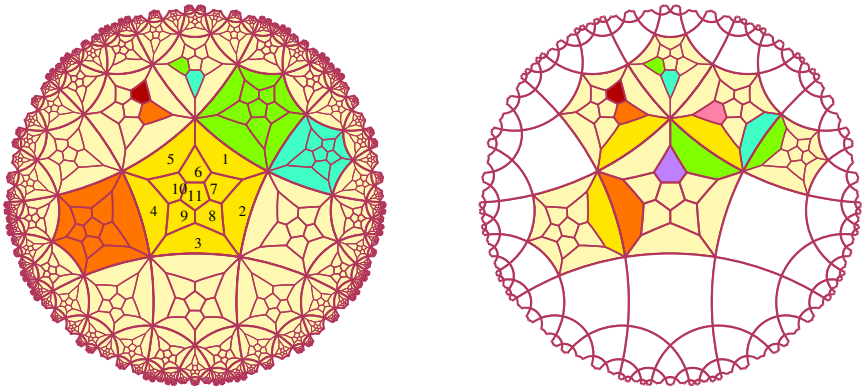


Fig. 2: Two different ways for representing a pseudo-projection on the pentagrid. On the left-hand side: the tiles have their colour. On the right-hand side: the colour of a tile is reflected by its neighbours only.

side of Figure 2, the tiles keep their colour.

We can see that this raises a problem with the tiles which are put on the faces 6 and 10 of the tile W, in case W would be blue, for instance. For: what colour should be that of face 6? Will it be the colour of W or the colour of the other dodecahedron which shares it with W? Another problem is given, for instance by the faces 1 of Y and G, assuming that the face 1 of G is that which sees Y. In fact, as can easily be seen by the fact that these faces are both perpendicular to Π_0 and that they share a common edge lying in Π_0, these faces coincide. The fact that they have different colour might be misleading.

This can be avoided by fixing a convention. In order to keep as much information as possible in the pseudo-projection, we shall consider that a face of a tile does not show the colour of the tile but the colour of its neighbour sharing the same face. The right-hand side of Figure 2 shows the same configuration as the one of the left-hand side, but under the new convention. Also, to make the figure more readable, we do not draw the pseudo-projection of a tile who would be white with only white neighbours among those of its neighbours which do not touch Π_0. Now we can see that we can use the fact that two different faces coincide in the $3D$ space by indicating the colour of the other tile.

Later on, we shall adopt this second solution to represent our cellular automaton. Indeed, the cells of the cellular automaton are the dodecahedra of the dodecagrid and the colour of a tile is given by the state

of the cellular automaton at the considered dodecahedron.

Before turning to the next section, we have an important remark.

We have already indicated that, in the pseudo-projection, faces which share an edge but which belong to different dodecahedra do coincide in the hyperbolic $3D$ space. The consequences are important with respect to the neighbours of a tile τ where, by neighbour, we mean a polyhedron which shares a face with τ. On the left-hand side part of Figure 2, we can see four small faces coloured with r, o, g and b on two white dodecahedra which we call W_1 and W_2, with W_1 being a neighbour of the central cell and W_2 a neighbour of the green neighbour of the central cell. We can view these coloured faces as dodecahedra obtained from the dodecahedron to which the face belongs by reflection in the very face. Call these dodecahedra by the colour of their defining faces. Considering the planes of the faces and their relations with Π_0, it is not difficult to see that the dodecahedra r and g are neighbours as well as the dodecahedra o and b. However, despite the fact that the corresponding faces share an edge, the dodecahedra r and o are not neighbours. However, as r, o and W_1 share a common edge, there is a fourth dodecahedron δ_1 sharing this edge which is not represented in the figure. Now, δ_1 plays an important role for both r and o as it is a neighbour for both of them. The same remark holds for the dodecahedra g and b for which there is a dodecahedron δ_2, a neighbour of both dodecahedra, sharing a common edge also with W_2. Moreover, it can be seen that δ_1 and δ_2 are also neighbours: their common face is in the plane of the common face of W_1 and W_2, which also contains the common face of r and g as well as the common face of o and b.

Both couples r with g and o with b can also be seen on the right-hand side part of Figure 2. There are also two other coloured small faces: a purple one on the central tile, call it p as well as the dodecahedron which it defines. There is also a pink one on the green dodecahedron which is a neighbour of the central one. Call the pink dodecahedron π. It is not difficult to see that the following pairs of dodecahedra are neighbours in the dodecagrid: b and π, π and p as well as p and o. Moreover, the four dodecahedra o, b, π and p share a common edge which belongs to the same line as the one which supports the edge shared by W_1, W_2, the central tile and G.

At last, remark that the plane of a face of a dodecahedron D defines two half-spaces: the half-space which does not contain D contains one neighbour of D exactly. The half-space which contains D contains all the other neighbours of D also. This can be seen as a consequence of the convexity of the dodecahedron.

Now, we can turn to the implementation of the railway model in the dodecagrid.

3 Implementation of the tracks

The implementation of the model is much more difficult in the hyperbolic $3D$-space than in the hyperbolic plane. Speaking about implementations in the hyperbolic plane, I often use the metaphor of a pilot flying with instruments only. This can be reinforced in the case of the hyperbolic $3D$-space by saying that this time we are in the situation of an astronaut who can do no other thing than fly with instruments only: sometimes, the astronaut may look at the earth. It is a fantastic image, however of no help for the navigation in cosmos. For the dodecagrid, we hope that the method explained in Section 2 shows that the situation is after all a bit better than in cosmos. The figures which we can obtain from the projections defined in Section 2 may help the reader to have a satisfactory view of the situation. We have to never forget that the views we can obtain are dramatically simplified images of what actually happens. However, always bearing in mind that the images are always a local view, a good training based on rigorous principles may transform them into an efficient tool.

Remember that in most its parts, the track followed by the locomotive runs on a fixed plane of the hyperbolic $3D$ space. We shall see that we can assume that this plane is Π_0. Only occasionally, it switches to other planes, perpendicular to Π_0. In particular, this is the case for the implementation of crossings: as in [7], we take advantage of the third dimension in order to replace them by bridges. Also for the sensors which decorate the switches, we shall take benefit of the third dimension to differentiate the configurations of the various switches.

In this section, we deal with the tracks only, postponing the implementation of the switches to Section 4.

3.1 The pieces

Below, Figure 3 illustrates a copy of the most common element of the tracks, which we call the **straight element**. It consists of a single dodecahedron, the track itself, marked by four blue dodecahedra, the **milestones**, which are neighbours of this dodecahedron.

Note the numbering of the faces on the figure: it follows the convention mentioned in Section 2. In Figure 3, pictures (a) and (b), face 0 is not visible but it is visible in the other pictures. Similarly, face 5 and face 2 respectively, are not visible in pictures (c) with (d) and (e) with (f) respectively. Due to the role of the elements in the circuit, we shall say that face 1 is the **entry** of the element and that faces 3 and 4 are its **exits** in the case of pictures (a) and (b). In the case of pictures (c) with (d) and (e) with (f) respectively, the entries are face 4 with face 10 and face 3 with face 8 respectively. We shall say **exit 3**, **exit 4**, **exit 8** or **exit 10** if we need to make it more accurate. It is important to notice that exits and entries can be exchanged: we can have exit 1 and entry 3 but not exit 3 and entry 4. Such a change of direction is necessary, but it will be realized by another element. As the role of entry and exits can be exchanged, we shall use the word **exit** in general descriptions with the possible meaning of both an entry or an exit through the possibly indicated face. Remember the convention we introduced in Section 2. In

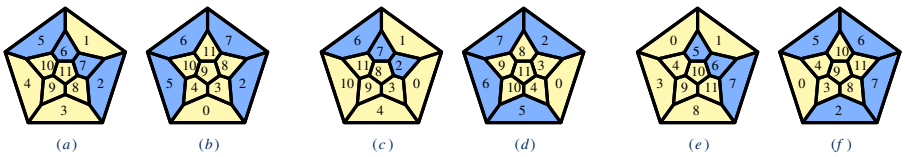


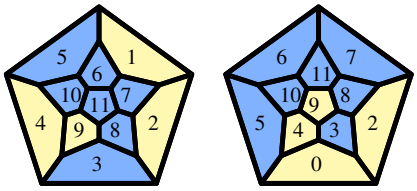
Fig. 3: An ordinary element of the track. Figure (a) is a view from above. Figure (b) is a view from the back of face 1. The locomotive enters the element via face 1 and exits via face 3 or face 4. It also may enter via face 3 or face 4 and then it exits through face 1.
In Figures (c) and (d), the element is a bit turned around face 1 and the exits are now face (4) and (10). In Figures (e) and (f), the element is turned around face (1) too, but in the opposite direction, and the exits are now faces 3 and 8. The motion in the opposite direction is always possible.

most figures of the paper, if not otherwise mentioned, the colour of a cell can be deduced from the colours of the face of its neighbours. As an example, in the pictures of Figure 3, the milestones are blue and they are neighbours of the element.

As the name suggests, the milestones are usually fixed elements: they are not changed by the passage of the locomotive. This means that the milestones always remain blue, while the track is white as most cells of the space itself: the white state plays the role of the quiescent state: if a cell is white as well as all its neighbours, then it remains white.

In Figure 3, the pictures represent various positions of the same elements which can be obtained from each other by a rotation a face of the dodecahedron or by a product of such rotations. We refer the reader to Subsection 5.1 where this problem is examined. In the figure, pictures (a) and (b) show a situation where Π_0 is the plane of face 0. In the pictures (c) and (d), it is that of face 5. In the pictures (e) and (f), it is that of face 2. The milestones can be viewed as the materialization of a catenary over the track itself, assumed to be put on the plane of the element.

Figure 4 illustrates another element of the track which we call a **corner**. This element allows the locomotive to perform a turn at a right angle. This possibility is very important and absolutely needed, as we shall see later.



(a) (b)

Fig. 4: The corner element of the track. Figure (a) is a view from above. Figure (b) is a view from the back of face 1. The locomotive enters the element via face 1 and exits via face 2. It also may enter via face 2 and then it exits through face 1.

As we can see from Figure 4, the corner has more milestones around it then a straight element: 7 milestones instead of 4 ones. However, the face of a corner on Π_0 is white, while for a straight element the face on Π_0 is usually blue.

3.2 Vertical and horizontal segments

When finitely many straight elements are put one after each other, with the entry of one of them shared by the exit of the previous one, we say that these elements are set into a **vertical segment**, **vertical** for short, provided that the plane of these elements is the same and that there is a line of this plane which supports one side of each element which we call the **guideline**. Figure 5 illustrates the basic example of a vertical. The guideline supports a side of the faces 0 of the elements and the common plane is that of the faces 5.

In the representation of Figure 5, the dodecahedra are projected on the plane of face 5.

In the left-hand side picture of the figure, number the elements of the figure from 1 to 7. We can see that the element i is in contact with the element i+1 with $i \in \{1..6\}$. Consider elements 3 and 4, the latter one occupying the central pentagon of the picture. The exit 4 of element 4 and the entry 1 of element 3 appear as different faces of dodecahedra: each one is projected inside the face 5 of the dodecahedron. Now, by definition, the entry 1 of element 3 and the exit 4 of the element 4 coincide. Indeed: elements 3 and 4 have their faces 5 on a common plane. They also have their sides 0 on the guideline. The entry 1 of element 3 and the exit 4 of element 4 are perpendicular to the guideline and they share a common side: they are the same face.

As we stressed in Section 2, this situation is important and we shall not repeat this point systematically. It is a property of the hyperbolic $3D$ space which we have to bear in mind while looking at the figures.

Note that in the figure, the entry 1 of an element is connected with the exit 4 of the previous one. Of

course, the segment can be run in the opposite direction: then an exit 4 becomes an entry 4 and an entry 1 becomes an exit 1.

In the righ-hand side picture of Figure 5, we represent another kind of track which we shall call **horizontal segments**. Such tracks consists of finitely many elements which can be written as a word of the form $(SeC)^k$, where Se denotes a straight element and C denotes a corner. The entry of the corner abuts an exit of the straight element. It is not always the same exit. In fact, there is an alternation of the exits which makes a Fibonacci word: if we associate to SeC the number of the exit of the straight element which abuts the entry of the corner, then this defines a homomorphism of $(SeC)^k$ on a factor of length k of the infinite Fibonacci word. Indeed, all corners are put on a black node of the Fibonacci tree. Straight elements are put on either white or black nodes. This can be made more accurate as follows. The straight elements of the segment are in contact of cells of the level n of the tree while the straight elements themselves are in the level n+1. The corners of the segment are all in the level n+2. Now, when the straight element is put on a white node, the exits are through faces 1 and 4. When it is put on a black node, the exits are through faces 1 and 10. This explains the connection of a horizontal segment with the infinite Fibonacci word, also see [6].

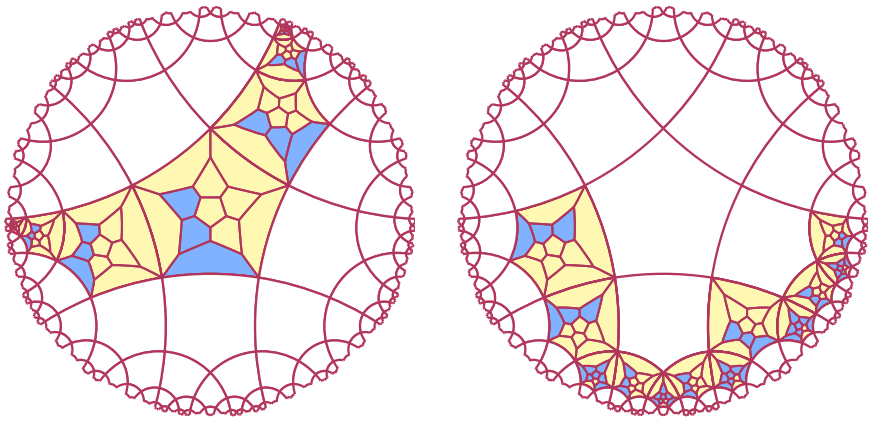


Fig. 5: Pseudo-projection on the plane of the track of its elements: left-hand side, case of a vertical segment; right-hans side, case of a horizontal one.

In the right-hand side picture of Figure 5, all the cells, the leftmost one excepted, constitute an illustration of a horizontal segment. Note that the leftmost element does not belong to the horizontal segment but it realizes the connection with a vertical segment.

3.3 Bridges

As already mentioned in this section, crossings of the planar railway circuit are replaced by bridges. We can arrange the crossing in such a way that two vertical segments V_0 and V_1 cross each other. Assume that V_0 will remain in the plane Π_0 of its faces 5 while V_1 will make a detour in the plane Π_1, perpendicular to Π_0, which contains the guideline of its projection onto Π_0. In Π_1 the track will follow a horizontal segment which will take the cells of two circles of cells in Π_1: at a distance 2 or 3 from the cell c_0 of V_0 which has a contact with both Π_0 and Π_1. Figure 6 represent such a bridge using two pseudo-projections:

one onto the plane Π_0, on the left-hand side of the figure, and the other onto the plane Π_1 on its right-hand side. We shall say that the projection onto Π_0 is the view from above and that the projection onto Π_1 is the frontal view, both ways of views referring to the bridge itself.

Let us have a closer look at the figures.

In the view from above, we can see two vertical segments: one goes from the right-up part of the figure to the left-bottom one. It can be easily recognized as a copy of the vertical segment illustrated by Figure 5. Here, it contains two tiles coloured with light brown. We shall call this track the top-down track. The other track goes from the left-upper part of the figure and goes to the right-bottom one. We shall call it the left-right track. We can see the guideline of the top-down track. It is the intersection of the planes Π_0 and Π_1.

Still in the view from above, we can see golden yellow marks on the light brown tiles and two green marks on the central tile. The golden marks indicate that the top-down track goes on these tiles. The green marks indicate the two piles of the bridge, the light brwon tile being their basement. Number the cells of the projection of the top-down track in the view from above from 1 to 7, 1 being the number of the topmost cell. Cell 4 is the central cell and it belongs to the left-right track: the top-down track follows a horizontal segment on Π_1 which can be seen in the frontal view, see the right-hand side part of Figure 6. The departure/arrival of the horizontal segment is defined by cells 6 and 2 which can be seen on both views. In the frontal view, the trace of Π_0 can easily be seen: it is the border between the coloured tiles and the others which remain blank, on the bottom part of the figure. There are seven coloured cells along this line in the frontal view: they are exactly the cells number from 1 to 7 in the view from above. In the frontal view, cell 6 is on the left-hand side.

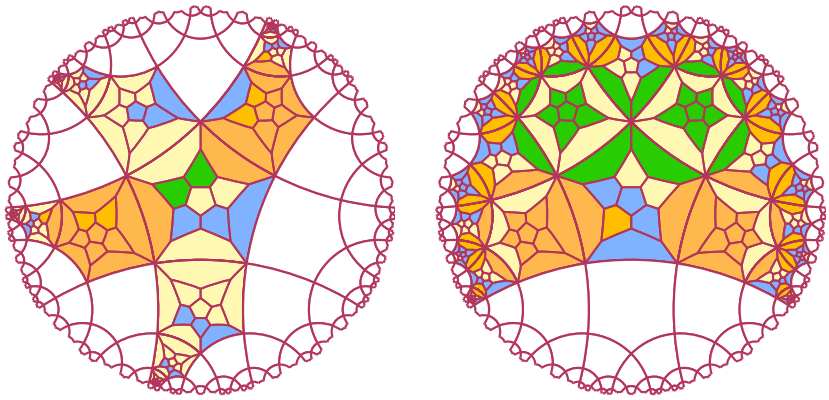


Fig. 6: Pseudo-projections of two tracks crossing through a bridge. On the left-hand side: pseudo-projection onto Π_0; on the right-hand side: pseudo-projection onto Π_1. The exit faces are marked by a golden yellow colour on the right-hand side figure. The piles of the bridge are marked with green on the central cell, their basement are marked with light brown. In the frontal view, the colour of the cell are exceptionally given to its faces which are not in contact with a cell of a track. This is to underline the elements of the bridge on which the track relies.

Cells 6 and 2 are an application of what we have mentioned in Subsection 3.1, about the different ways to rotate a straight element in order to access to another plane. Cell 6 is alike the picture (c) of Figure 3. Its faces 0 and 5 are in contact with the guideline. Now, exit 3 may be used to go into the plane of face 0

which is perpendicular to the plane of face 5. This will be the starting point of our bridge. Note that face 0 of cell 6 is on Π_1. From this tile, the bridge follows a horizontal segment until it arrives at cell 2 which is in contact with both Π_0 and Π_1. Notice that cell 2 is alike the picture (e) of Figure 3, and that its face 4 is the face where the track of the bridges again joins the top-down track. Note that in cell 2, face 0 is on Π_1 as this is the case for cell 6. Looking at the cells of the horizontal segment in the frontal view, we can notice that the straight elements have a milestone which is below Π_1: this means that there are milestones on both half-spaces defined by Π_1. Now, for the corners, all the milestones are in a same half-space defined by Π_1. For the straight elements, their exits are most often the faces 1 and 4. However, from time to time, the exits are the faces 1 and 10. In Subsection 3.2 we have seen the reason of these variants. As can be seen on the frontal view, all corners are put on a black node of the Fibonacci tree and straight elements can be either on a black node or on a white one. From Subsection 3.2, we know that when a straight element is on a white tile, the exits are the faces 1 and 4. When it is on a black tile, it is the faces 1 and 10.

3.4 The motion of the locomotive on the tracks

Presently, we describe the motion of the locomotive on its tracks. We refer the reader to [11] for figures for all the possible motions on the tracks and through a switch.

This motion is very different from the simulation of [7]. There, the tracks were materialized by a specific colour and the locomotive simply occupied two contiguous cells of the track. Here, if we consider the track as constituted of the blank cells surrounded by milestones as in [9, 14, 13, 10], then the motion of the locomotive is very similar. In particular, it is exactly the same as the planar simulation described in [10]. Accordingly, restricting our attention to the cells of the track, we have the following one-dimensional rules for the motion of the locomotive:

```
B W W → B        W W B → B
R B W → R        W B R → R
W R B → W        B R W → W
W W R → W        R W W → W
```

As can easily be deduced from the rules, the locomotive consists of two contiguous cells: one is blue, the front, the other is red, the rear.

With the just mentioned principles in mind, we can easily device the rules for the motion of the locomotive. See [11] for the systematic writing of the corresponding rules and for illustrative figures.

From the rules we can devise, it is worth noticing that the elements can be freely assembled, provided that they observe the principle which we have fixed: exits of an element are 1 and 2 for corners, they are 1 and 3, 1 and 4, 1 and 8 or 1 and 10 for a straight element. Other combinations are ruled out by the rules.

4 Implementation of the switches

In order to describe the switches, we shall focus on the memory switch which has the most complex mechanism among the switches. In fact, this mechanism consists of two connected parts A and B. In the study of the other switches, we shall see that fixed switches use mechanism A alone and that the flip-flop switches use mechanism B alone.

All switches will share the following common features. They are assumed to be on the same plane Π_0. However, certain parts of the above mechanisms are on both half-spaces defined by Π_0. This is why we

shall present two figures for each switch: one is a pseudo-projection from above onto Π_0, the other is a pseudo-projection onto the same plane, but from below. We have to remember that in such a case, the left-hand side and the right-hand side are exchanged as well as clockwise and counter-clockwise motions.

Next, for two of them, the switches have both a left- and a right-hand side version. In the left-, right-hand side version respectively, the active passage sends the locomotive to the left-, right-hand side track respectively. However, for the fixed switch, a left-hand side version is enough. A right-hand side fixed switch is obtained from a left-hand side one as follows: after the switch, the left-hand side track crosses the right-hand side one in order to exchange the directions. Thanks to the bridge which we have implemented, this is easily performed.

The switches will be presented according to a similar scheme.

First, we describe what we call the **idle configuration**: it is the situation of the switch when it is not visited by the locomotive. All switches are the meeting point of three tracks. The meeting tile is a straight element and, in the figures, which will represent idle configurations only, it is placed at the central tile. The track which arrives to the entry 1 of this element represents the arrival for an active crossing of the switch. Exit 3 gives access to the track which goes to the left and exit 4 gives access to the track going to the right. In the computer program used to check the simulation, the cells of the tracks are numbered from 1 to 11 and from 12 to 16. In the figures, we can see cells 2 to 10 and 12 to 15 only. Cells 1 to 5 constitute the arriving track. They follow a vertical segment which arrives to the leading tile of a quarter constructed around the central cell. We shall number this sector by 1, as the exit to which the track leads. Cell 2 is the farthest visible cell from the central cell, cell 5 is the leading tile of sector 1. The central cell is cell 6. Cells 7 to 10 constitute the track which leaves the switch through exit 3. They are displayed in a vertical segment included in a sector lead by cell 7 and which is called sector 3, after exit 3. Cells 12 to 15 constitute the vertical segment which leaves the switch through exit 4. These cells belong to sector 4 headed by cell 12, see Figure 7 for instance.

A closer look shows that the tracks are not exactly along a vertical: the cell which is in contact with an exit of the central cell, is a straight element whose face 0 is on Π_0. The next cell, cell 4, 8 and 13 respectively is a corner, again with its face 0 on Π_0. The remaining two cells constitute a vertical segment in the way we have defined them with a milestone below Π_0 with respect to the other milestone which we consider as upon this plane.

With these conventions, we can start the study of each switch. We shall see the memory switches, the fixed switch and the flip-flop switches in this order.

4.1 Memory switches

As mentioned in the beginning of this section, the memory switches are the most complex construction in our implementation.

In the paper, we represent the left-hand side memory switch only, see Figure 7. The reader can see the figure corresponding to the right-hand side memory switch in [11].

In the figure, there is a big disc and a smaller one. The big disc is a pseudo-projection onto Π_0 from above, while the smaller one is a pseudo-projection onto the same plane from below. In both discs, we apply the convention about the colour of the cells.

In the memory switch, there are two **sensors**, two **markers** and two **controllers**. The sensors are cells 17 and 18 which are neighbours of the cells 7 and 12 respectively through their faces 0. Cells 7 and 12 are called the **scanned cells**, inspected by their sensors. The **upper** controller is cell 20 which is the second common neighbour of cells 7 and 12 above Π_0: the first common neighbour is the central cell.

We consider that cell 20 has its face 0 on Π_0. The **lower** controller is cell 19 which is the neighbour of cell 20 through its face 0: cell 19 is thus below Π_0 and we also consider that its face 0 is on Π_0. The two markers are cells 21 and 22: they are neighbours of cell 20 through its faces 8 and 10 respectively. Now, the sensor of cell 7 is blue and that of cell 12 is red. Similarly, cell 21 is red and cell 22 is blue. The colours of the sensors and of the markers allow to identify the left-hand side memory switch. In a right-hand side memory switch, the colours of the sensors and the markers are exchanged: cells 21 and 18 are blue, cells 22 and 17 are red.

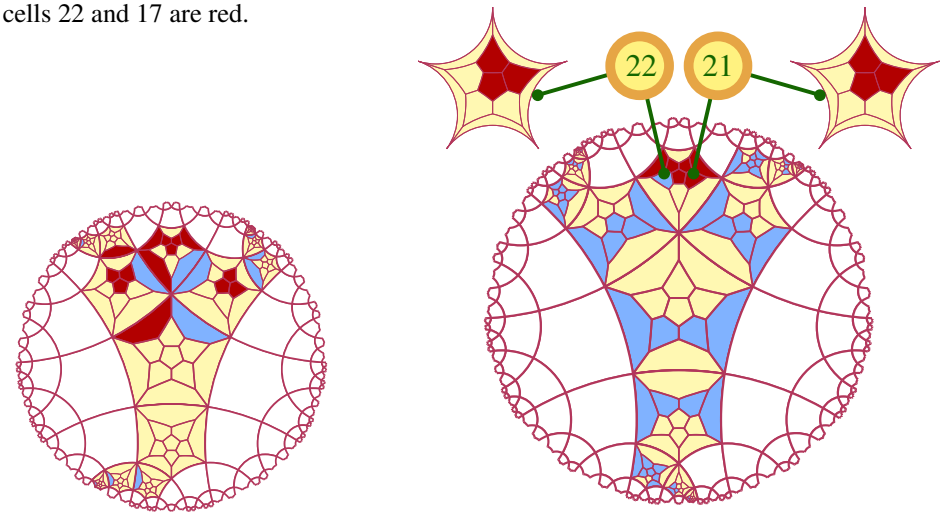


Fig. 7: The idle configuration of a left-hand side memory switch, represented by the two pseudo-projections, one from above: the big disc; the other from below: the small disc.

The working of the memory switch is the following.

A blue sensor is indifferent to the direction of the locomotive: it may cross the cell it scans in both ways. A red sensor does not behave the same. First, it prevents the locomotive to enter the cell it scans in an active passage. In a passive passage, it detects the passage of the locomotive, it allows it to pass through the cell it scans, but it reacts to the passage by changing its colour: as the blue sensor does not see the red one, the red sensor cannot change its colour to blue. It changes it to white. This is detected by the lower controller, usually blue, which becomes red. When the lower controller is red, both sensors change their colour: the blue one to red the now white one to blue. And the lower controller goes back to blue. Now, the upper controller, usually blue, also detects the passive passage through the non-selected track: its markers allow it to differentiate cell 7 from cell 12. And so, when the front of the locomotive leaves cell 7 or cell 12 when this cell is on the non-selected track, the upper controller becomes white and then red. It becomes white to prevent the locomotive from being duplicated on the selected track: the locomotive must go through entry 1. Then it becomes red, at the same times as the lower controller becomes red. When the upper controller is red, both markers exchange their colour and at the next time, the upper controller returns to blue.

We have no room in this paper for figures about the motion of the locomotive. We refer the reader to [11] for such figures. In that document, the reader may also find tables of the execution of the computer program which simulated the various motions. For the memory switch, we give one such table here: the

Tab. 1: Run of the simulation program. A corresponding figure can be found in [11]. The passive crossing through the non-selected track correspond to cells 12 up to 16, in the reverse order and then to cells 1 up to 6 in the reverse order too.

```
passive crossing of a memory switch, left-hand side,  through the NON selected track:
          1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
time 0 :  W  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  B  R  B  B  R  B
time 1 :  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  W  B  R  B  B  R  B
time 2 :  W  W  W  W  W  W  W  W  W  W  W  B  R  W  W  W  B  R  B  B  R  B
time 3 :  W  W  W  W  W  B  W  W  W  W  W  R  W  W  W  W  B  W  B  W  R  B
time 4 :  W  W  W  W  B  R  W  W  W  W  W  W  W  W  W  W  B  W  R  R  R  B
time 5 :  W  W  W  B  R  W  W  W  W  W  W  W  W  W  W  W  R  B  B  B  B  R
time 6 :  W  W  B  R  W  W  W  W  W  W  W  W  W  W  W  W  R  B  B  B  B  R
time 7 :  W  B  R  W  W  W  W  W  W  W  W  W  W  W  W  W  R  B  B  B  B  R
```

table which corresponds to the motion of the locomotive when it passively crosses the switch from the non-selected track, see Table 1.

In a heading line, the trace indicates the visited cells by their number as well as their immediate neighbours, also numbered, when they may be changed during the visit. Below this leading line, for each time, the state of a cell is given in the column corresponding to its number. Looking at the numbers, we can see that the simulation program considered two more cells on the track arriving to the switch than what is shown by the figure. In this table, we can see that the sensors and the markers play an active role and, at the end of the role, they are exchanged. Accordingly, the locomotive entered a left-hand side memory switch and it leaves a right-hand side memory switch.

Note that Table 1 shows exactly when each sensor and controller is triggered. The front of the locomotive is in cell 12 at time 2. This makes cell 20 and 18 becoming white. As already noticed, the red sensor cannot change to blue as the blue sensor, which cannot see neither cell 12 nor cell 18, did not yet realized that a change must occur. At time 3, the front of the locomotive is now in cell 6, the central cell, and cells 20 and 18 are now white. This is the signal for both controllers to flash the red signal which will trigger the exchange of colours in the sensors and in the markers. The signal is sent at time 4 and the exchange of colours happens at time 5: starting from that time, the memory switch is now a right-hand side one.

In [11], it can be checked that the right-hand side memory switch reacts in a similar way to its passive crossing by the locomotive through the non-selected track. In particular, when the locomotive leaves the switch, it is now a left-hand side one.

4.2 Fixed switches

Figure 8 illustrates the idle configuration of a fixed switch. As announced at the beginning of Section 4 we can see on the figures that the idle configuration of a fixed switch is, in some sense the half of the configuration of a left-hand side memory switch. The point is that there is no lower controller and that the sensors and markers are now fixed milestones. Two of them, cell 18 and 21, are always red and the others, cell 17 and 22 are always blue. Now, the fixed switch keeps the upper controller. As already noticed, the red milestone prevents the locomotive to go through the non-selected track in an active passage. However, as also noticed in the study of memory switches, the change of colour in the sensor of cell 12 is not enough

to prevent the locomotive to go from the central cell both to cell 5, as required, and to cell 7 which should be avoided. Cell 7 cannot itself prevent such a passage because it sees cell 6 but it does not see at the same time cell 12. Now, we have seen in Subsection 4.1 that the upper controller is able to perform this tasks: as soon as it sees that the front of the locomotive is in cell 12, it becomes white. As cell 7 sees this new colour at the same time when the front of the locomotive is in cell 6, it allows it to reject the access of the locomotive to the selected track.

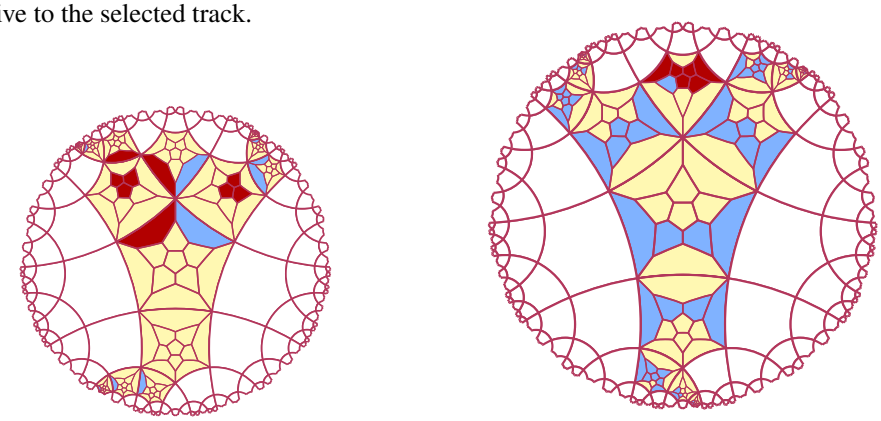


Fig. 8: The idle configuration of a fixed switch. The big disc is a view from above, the small one a view from below.

It is not difficult to check that Figure 8 implements this changes. It was just enough to neutralize the lower controller by changing its colour from blue to blank. Moreover, the cell itself has no other non-blank neighbour than the sensors. Similarly, as the sensors and markers are fixed milestones, this means that their neighbours are all blank, except the cell of the switch with which they are in contact: cell 7 or 12 for the sensors, cell 20 for the markers. Consequently, the configuration of the fixed switch is a bit simpler than that of the left-hand side memory switch. It also requires less non-blank cells.

Tab. 2: Run of the simulation program corresponding to the passive crossing through the non selected track. The corresponding cells are cell 12 up to 16, in the reverse order and then cells 1 up to 6 in the reverse order too.

```
passive crossing of a fixed switch, NON selected track :
          1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
time 0 :  W  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  B  R  W  B  R  B
time 1 :  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  W  B  R  W  B  R  B
time 2 :  W  W  W  W  W  W  W  W  W  W  W  B  R  W  W  W  B  R  W  B  R  B
time 3 :  W  W  W  W  W  B  W  W  W  W  W  R  W  W  W  W  B  R  W  W  R  B
time 4 :  W  W  W  W  B  R  W  W  W  W  W  W  W  W  W  W  B  R  W  R  R  B
time 5 :  W  W  W  B  R  W  W  W  W  W  W  W  W  W  W  W  B  R  W  B  R  B
time 6 :  W  W  B  R  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  B  R  B
time 7 :  W  B  R  W  W  W  W  W  W  W  W  W  W  W  W  W  B  R  W  B  R  B
```

We refer the reader to [11] for figures illustrating the three possible crossings of the switch by the locomotive. There, each figure is accompanied by a table which shows a trace of the execution of the

simulating program corresponding to that crossing. Here, we reproduce Table 2 only which gives the trace of a passive crossing through the non-selected track.

Table 2 also allows us to check that a half-control, namely that of cell 20 was enough to guarantee the correct working of the switch. It also shows that it was enough to block the changing of sensors by transforming them into milestones. This is an interesting point which shows us another advantage which we can take from the third dimension.

Indeed, in previous simulations in the hyperbolic plane on the heptagrid, with six or four states, we had a curious phenomenon during the active passage of the locomotive and also during a passive crossing for the fixed switch too. In these simulations, the passage of the locomotive created a duplicate of its front towards the wrong direction. However, as this new front was not followed by a red rear, it was possible to erase it, simply by appending a few rules.

4.3 Flip-flop switches

Figure 9 show the idle configuration of the left-hand side flip-flop switches. The corresponding figure for right-hand side switches can be seen in [11]. As announced at the beginning of Section 4 we can see on the figures that the idle configuration of a flip-flop switch switch is, in some sense the half of the configuration of a memory switch of the same laterality. The point is that there is no upper controller and, consequently, no markers. However, the sensors and the lower controller are still present. The lower controller is exactly the same as in the memory switches and it works in the same way. Contrarily to the fixed switch, the sensors are not milestones. They are true sensors like in the memory switch.

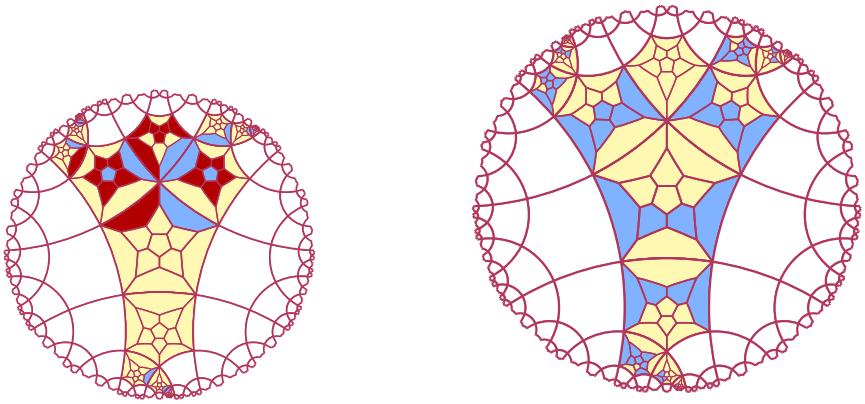


Fig. 9: The idle configuration of the left-hand side flip-flop switch. The big disc is a view from above, the small one, a view from below.

However, they are a bit different from the sensors of the memory switch as they work in a different way. The difference can be noticed in the small discs of Figures 7 and Figures 9. In Figures 7, the sensors are marked by a group of three red milestones, on pairwise contiguous faces, one of them being the face which is opposite to that in contact with the scanned cell. In Figures 9, the sensors are marked by a ring of five milestones whose contact faces with the sensor are around the face which is opposite to the face in contact with the scanned cell. Also, the face opposite to that which is shared with the scanned cell is blue as it is in contact with a blue milestone. Similar figures for the right-hand side memory or flip-flop

Tab. 3: Run of the simulation program. A corresponding figure can be found in [11]. The active passage visits the cells of the selected track: cells 1 up to 11 in this order.

```
active crossing of a left-hand side flip-flop switch :
          1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
time 0 :  W  R  B  W  W  W  W  W  W  W  W  W  W  W  W  W  B  R  B  W  W  W
time 1 :  W  W  R  B  W  W  W  W  W  W  W  W  W  W  W  W  B  R  B  W  W  W
time 2 :  W  W  W  R  B  W  W  W  W  W  W  W  W  W  W  W  B  R  B  W  W  W
time 3 :  W  W  W  W  R  B  W  W  W  W  W  W  W  W  W  W  B  R  B  W  W  W
time 4 :  W  W  W  W  W  R  B  W  W  W  W  W  W  W  W  W  B  R  B  W  W  W
time 5 :  W  W  W  W  W  W  R  B  W  W  W  W  W  W  W  W  W  R  B  W  W  W
time 6 :  W  W  W  W  W  W  W  R  B  W  W  W  W  W  W  W  W  R  R  W  W  W
time 7 :  W  W  W  W  W  W  W  W  R  B  W  W  W  W  W  W  R  B  B  W  W  W
```

switches can be found in [11].
This difference is explained by the fact that the working of the sensors is very different from that of the memory switch, quite the opposite: in the memory switch, the blue sensor is passive and the red sensor blocks the access to the non-selected track in the active passage and turns to white when the front of the locomotive appears in the scanned cell in a passive crossing. In the flip-flop switch, the red sensor only blocks the access to the non-selected track and the blue sensor is active: when the front of the locomotive leaves the scanned cell, it becomes white, triggering the flash of the lower controller at the next time which, to its turn, makes the sensors exchange their colour.

This can be checked in Table 3 and similar tables of [11]. The front of the locomotive is in the scanned cell at time 4, so that the blue sensor is white at time 5. As in the case of the memory switch, as cell 17 and 18 do not see each other, the blue sensor cannot turn to red immediately. It becomes white which triggers the flash of the controller at time 6 and the exchange of colours between the sensors at time 7 only.

5 About the rules and the computer program

We have no room for the rules which are displayed in [11]. However, we have a preliminary work on rotation invariance which is by itself interesting.

In order to write the rules of the cellular automaton, we shall use the numbering of the faces of a dodecahedron which was mentioned in Section 2 and which was used in Sections 3 and 4. However, there was no fixed rule to connect the numbering of a cell to that of a neighbouring one except for the cells of the track, as we did in Subsection 3.2. This is not a big problem as, in fact, the rules which we shall devise have an important property: they are **rotation invariant**, which means that they are not changed by a motion which leaves the dodecahedron globally invariant and which preserves orientation.

In the plane, the characterization of rotation invariance in the rules is easy to formulate: it is necessary and sufficient that the rules are not changed by a circular permutation on the neighbours. In the case of the pentagrid, this means that once we fixed a rule, we automatically append to the table of rules the other four permuted images of the rule. Here, the characterization is far less trivial. In [7], we could avoid this problem by imposing a stronger condition on the rules, namely to be **strongly lexicographically** different from each other. This means that to each rule, we associate a word of the form $A_1^{k_1}..A_n^n$ where A_1, ..., A_n

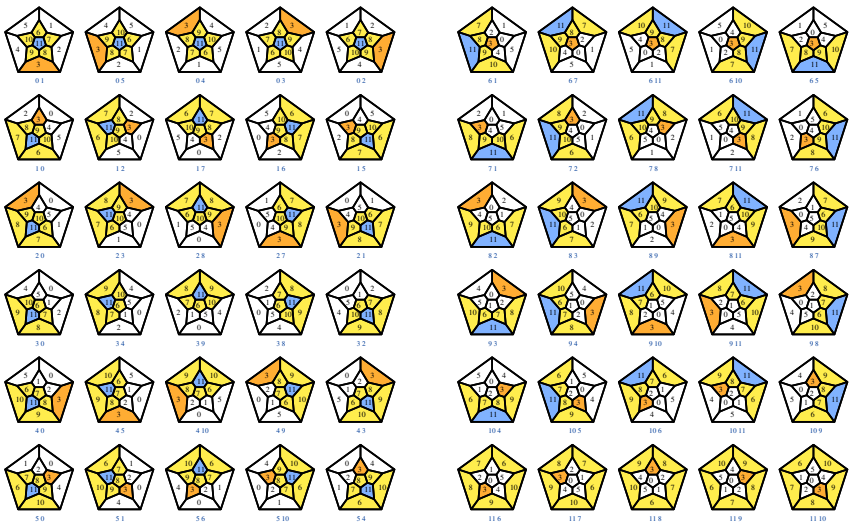


Fig. 10: The map of the positive motions leaving the dodecahedron globally invariant.

are the states and k_1 are non-negative numbers satisfying $k_1 + ... + k_n = v$+1, where v is the number of neighbours of the cell, the cell being not counted. This was possible with 5 states and I could not keep this condition for 3 states. This is why we first study how to check rotation invariance for our cellular automaton in the hyperbolic $3D$ space.

5.1 Rotation invariance

The question is the following: how does a motion which leaves the dodecahedron globally invariant affect the numbering of its faces, an initial numbering being fixed as in Section 2?

In fact, it is enough to consider motions which preserves the orientation, we shall say **positive** motions. As such a motion leaves the dodecahedron globally invariant, it transforms a face into another one. Accordingly, fix face 0. Then its image, say f_0, can be any face, face 0 included. Next, fix a second face which shares an edge with face 0, for instance face 1. Then its image, say f_1, is a face which shares an edge with f_0. It can be any face sharing a face with f_0. Indeed, let f_2 be another face sharing an edge with f_1. Then, composing the considered positive motion with a rotation around f_0 transforming f_1 into f_2, we get a positive motion which transforms $(0, 1)$ into (f_0, f_2). This proves that we get all positive motions leaving the dodecahedron globally invariant, by first fixing the image f_0 of face 0 and then by taking any face f_1 sharing an edge with f_0. Note that once f_0 and f_1 are fixed, the images of the other faces are fixed, thanks to the preservation of the orientation. Accordingly, there are 60 of these positive motions and the argument of the proof shows that they are all products of rotations leaving the dodecahedron globally invariant.

Figure 10 gives an illustrative classification of all these rotations. The upper left picture represents the image of a Schlegel diagram of a dodecahedron with the notation introduced in Section 2. Each image represents a positive motion characterized by the couple of numbers under the image: it has the form f_0 f_1,

Tab. 4: The faces around a given face.

	1	2	3	4	5
0	1	5	4	3	2
1	0	2	7	6	5
2	0	3	8	7	1
3	0	4	9	8	2
4	0	5	10	9	3
5	0	1	6	10	4
6	1	7	11	10	5
7	1	2	8	11	6
8	2	3	9	11	7
9	3	4	10	11	8
10	4	5	6	11	9
11	6	7	8	9	10

where f_0 is the image of face 0 and f_1 is the image of face 1. The figure represents two sub-tables, each one containing 30 images. Each row represents the possible images of f_1, f_0 being fixed. The image of face 0 is the plane of projection of the dodecahedron. The image of face 1 takes the place of face 1 in Figure 1. As an example, $f_0 = 0$ for the first row of the left-hand side sub-table, and in the first row, the first image gives $f_1 = 1$, so that it represents the identity. The other images of the row represent the rotations around face 0.

The construction of Figure 10 was performed by an algorithm using Table 4. For each face of the dodecahedron, the table gives the faces which surround it in the Schlegel diagram, taking the clockwise order when looking at the face from outside the dodecahedron, this order coinciding with increasing indices in each row. This coincides with the usual clockwise order for all faces as in Figure 1, except for face 0 for which the order is counter-clockwise when looking above the plane of the projected image. The principle of the drawings consists in placing f_0 onto face 0 and f_1 onto face 1. The new numbers of the faces are computed by the algorithm as follows. Being given the new numbers f_0 and f_1 of two contiguous faces φ_0 and φ_1 in the Schlegel diagram, the algorithm computes the position of φ_1 as a neighbour of φ_0 in the table. This allows to place f_1 on the right face. Then, the algorithm computes the new numbers of the faces which are around φ_1 in the table: it is enough to take the position of φ_0 as a neighbour of φ_1 and then to turn around the neighbours of f_1, looking at the new numbers in the row f_1 of the table, starting from the position of f_0. This gives the new numbers of the faces which surround face 1. It is easy to see that we have all faces of the dodecahedron by turning around face 1, then around face 5, then around face 7 and at last around face 8. As in these steps, each round of faces starts from a face whose new number is already computed, the algorithm is able to compute the new numbers for the current round of faces, using Table 4 to find the new numbers. Let us call this algorithm the **rotation algorithm**.

Thanks to the rotation algorithm, it is easy to compute the **rotated forms** of a rule of the cellular automaton.

Let $\underline{\eta}\eta_0...\eta_{11}\underline{\eta'}$ be a rule of the automaton. In this format, η is the current state of the cell and $\eta(i)$ is the state of the neighbour through face i, also called neighbour i, and η' is the new state of the automaton. Remember that the current state of a cell is its sate at time t and that its new state is its state at time t+1.

Call $\underline{\eta}\eta_0...\eta_{11}$ the **context** of the rule. Let μ be a positive motion leaving the dodecahedron globally invariant. The **rotated form** of the rule defined by μ is $\underline{\eta}\eta_{\mu(0)}...\eta_{\mu(11)}\underline{\eta'}$ and, similarly, $\underline{\eta}\eta_{\mu(0)}...\eta_{\mu(11)}$ is the **rotated form** by μ of the context of the initial rule. We say that the cellular automaton is **rotation invariant** if and only two rules having contexts which are rotated forms of each other always produce the same new state.

Now, thanks to our study, we have a syntactic criterion to check this property. We fix an order of the states. Then, for each rule, we compute its **minimal form**. This form is obtained as follows. We compute all rotated forms of the rule and, looking at the obtained contexts as words, we take their minimum in the lexicographic order. The minimal form of a rule is obtained by appending its new state to this minimum. Now it is easy to see that:

Lemma 1 *A cellular automaton on the dodecagrid is rotation invariant if and only if for any pair of rules, if their minimal forms have the same context, they have the same new state too.*

Now, checking this property can easily be performed thanks to the rotation algorithm.

We refer to [11] for the detailed study of the rules. The rules given there have the property that the minimal forms of their contexts are pairwise distinct.

5.2 About the computer program

As indicated in the introduction, I wrote a computer program in order to check the correctness of the rules. The program was written in $ADA95$ and it implements the algorithms mentioned in the paper.

Before giving a short account on the program itself, I would like to stress that using a simulation program for this purpose is mandatory. The computations are so complex for a man, at least for me, that the help of the computer allows me to check that the rules are correct. What is meant by this latter expression? We mean two things: a syntactical one and a semantic one. The syntactical correctness is that there is no pair of rules with the same minimal context giving rise to different new states. This is the minimal condition when working with deterministic cellular automata, which is of course the case here. In this work, we reinforced the condition by checking that two rules with the same minimal context always give rise to the same new state: this guarantees that the automaton is not only correct, but that it is also rotation invariant.

Now, the semantic correctness means that the rules do what we expect from them to do. This is far more complex to check and this cannot be completely ascertained by proof. Again, the computer program is useful in this regard. We can implement the simulation in the program and then run it. If everything goes smoothly through, we can believe that the implementation is correct. There is no guarantee of that. There is no automatic checking that the implementation is a correct hyperbolic implementation. There is also no proof that the program itself is correct. However, the setting is rather involved and while adjusting the program, many errors in my first table of rules were found by the program. As an example, the program also indicated me the need to cover face 11 of cells 7 and 12 with a blue milestone: otherwise, the set of rules would not be rotationally invariant.

About the program implementation itself.

First, I implemented the algorithms to compute the minimal form of a rule and, taking advantage of this implementation, the program computed the PostScript program for Figure 10. All traces given in the tables of Section 4 were computed during the execution of the program by the program itself.

Each test of a crossing of the switch by the locomotive was performed within the same implementation frame: the cells of the tracks were gathered in a table of tables. The big table has 22 entries corresponding

to the numbering of the cells explained in Section 4. For each index of the big table, a table of 14 entries gives various information on the cell in its current state, and its neighbours. The neighbours correspond to faces and are numbered from 0 up to 11 as in the paper. For each face, it is indicated whether the neighbour through the face has a permanent state or a variable one. As an example, a milestone seen from a face of a cell of the track is permanently blue. When the neighbour has a variable state, the table indicates the index of this neighbour in the big table. In fact, the big table is first a list of the variable cells and, at this occasion, it collects a useful information about each cell. The program also computes a bigger trace where at each time, the big table is dispatched in full detail. Table 5 gives two short pieces of this trace, taken during an active passage of the locomotive, at time 1. We can see the information which was just indicated, v meaning 'variable' and f meaning 'fixed'. The other indications are self-explaining.

Tab. 5: Two pieces of the big trace of the program, corresponding to the simulation of an active passage of the locomotive through a left-hand side memory switch, at initial time.

```
 6   -1  0  1  2  3  4  5  6  7  8  9 10 11
      W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  v  v  f  f  f  f  f  f  f
            5     7 12
 7    W  B  W  B  W  W  B  B  B  W  W  W  B
      v  v  v  f  f  v  v  f  f  f  f  f  f
        17  6        8 20
 8    W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  f  v  f  f  f  f  f  f  f
            7        9
 9    W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  f  v  f  f  f  f  f  f  f
            8       10
10    W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  f  v  f  f  f  f  f  f  f
            9       11
11   -1  0  1  2  3  4  5  6  7  8  9 10 11
      W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  f  f  f  f  f  f  f  f  f
           10
12    W  R  W  B  W  W  B  B  B  W  W  W  B
      v  v  v  v  f  v  f  f  f  f  f  f  f
        18  6 20    13
13    W  W  W  B  W  W  B  B  B  W  W  W  W
      v  f  v  f  f  v  f  f  f  f  f  f  f
           12       14
```

```
17    B  W  W  B  W  W  W  W  W  W  R  R  R
      f  v  f  v  f  f  f  f  f  f  f  f  f
         7     19
18    R  W  W  W  W  W  B  W  W  R  R  W  R
      f  v  f  f  f  f  v  f  f  f  f  f  f
        12              19
19    B  B  W  R  B  R  R  R  R  W  W  W  R
      f  v  f  f  v  v  f  f  f  f  f  f  f
        20        17 18
20    B  B  W  R  W  W  R  R  R  R  W  B  R
      f  v  f  f  v  v  f  f  f  v  f  v  f
        19        12  7           21    22
21   -1  0  1  2  3  4  5  6  7  8  9 10 11
      R  B  W  W  W  W  W  W  W  W  R  R  R
      v  v  f  f  f  f  f  f  f  f  f  f  f
        20
22    B  B  W  W  W  W  W  W  W  W  R  R  R
      v  v  f  f  f  f  f  f  f  f  f  f  f
        20
```

As the program performed a successful execution of all possible crossings and also along various vertical and horizontal segments with some mix of them, we can conclude that the proof of Theorem 1 is complete. □

References

[1] M. Cook. Universality in elementary cellular automata, *Complex Systems*, (2004), **15**(1), 1-40.

[2] F. Herrmann, M. Margenstern, A universal cellular automaton in the hyperbolic plane, *Theoretical Computer Science*, (2003), **296**, 327-364.

[3] M. Margenstern, Implementing Cellular Automata on the Triangular Grids of the Hyperbolic Plane for New Simulation Tools, **ASTC'2003**, (2003), Orlando, March, 29- April, 4.

[4] M. Margenstern, The tiling of the hyperbolic $4D$ space by the 120-cell is combinatoric, *Journal of Universal Computer Science*, **10**(9), (2004), 1212-1238.

[5] M. Margenstern, Two railway circuits: a universal circuit and an NP-difficult one, *Computer Science Journal of Moldova*, **9**, 1–35, (2001).

[6] M. Margenstern, Tilings of hyperbolic spaces: the splitting method and group theory, **WORDS'2003**, TUCS General Publications, **43**, (2003), 31-35.

[7] M. Margenstern, A universal cellular automaton with five states in the 3D hyperbolic space, *Journal of Cellular Automata* **1**(4), (2006), 315-351.

[8] M. Margenstern, Cellular Automata in Hyperbolic Spaces, Volume 1, Theory, *OCP*, Philadelphia, (2007), 422p.

[9] M. Margenstern, Cellular Automata in Hyperbolic Spaces, Volume 2, Implementation and computations, *OCP*, Philadelphia, (2008), 360p.

[10] M. Margenstern, A universal cellula automaton on the heptagrid of the hyperbolic plane with four states, *Theoretical Computer Science*, (2010), *doi:*10.1016/j.tcs.2010.04.015.

[11] M. Margenstern, A weakly universal cellular automaton in the hyperbolic $3D$ space with three states, *arXiv*:1002.4290[cs.DM], (2010), 54pp.

[12] M. Margenstern, G. Skordev, Tools for devising cellular automata in the hyperbolic 3D space, *Fundamenta Informaticae*, **58**, N°2, (2003), 369-398.

[13] M. Margenstern, Y. Song, A universal cellular automaton on the ternary heptagrid, *Electronic Notes in Theoretical Computer Science*, **223**, (2008), 167-185.

[14] M. Margenstern, Y. Song, A new universal cellular automaton on the pentagrid, *Parallel Processing Letters*, **19**(2), (2009), 227-246.

[15] D.M.Y. Sommerville, An introduction to the geometry of N dimensions, Dover Publ. Inc., New-York, 1958.

[16] I. Stewart, A Subway Named Turing, Mathematical Recreations in *Scientific American*, (1994), 90-92.

[17] S. Wolfram. A new kind of science, *Wolfram Media, Inc.*, (2002).