A Weakly Universal Cellular Automaton in the Pentagrid with Five States

Maurice Margenstern(✉)

Université de Lorraine, LITA, EA 3097 Campus du Saulcy,
57045 Metz, Cédex 1, France
margenstern@gmail.com

Abstract. In this paper, we construct a cellular automaton on the pentagrid which is planar, weakly universal and which have five states only. This result much improves the best result which was with nine states.

Keywords: Cellular automata · Universality · Tilings · Hyperbolic geometry

1 Introduction

In this paper, we construct a weakly universal cellular automaton on the pentagrid, see Theorem 2 at the end of the paper. Two papers, [1,7] already constructed such a cellular automaton, the first one with 22 states, the second one with 9 states. In this paper, the cellular automaton we construct has five states only. It uses the same principle of simulating a register machine through a railway circuit, but the implementation takes advantage of new ingredients introduced by the author in his quest to lower down the number of states, see [5]. The reader is referred to [3–5] for an introduction to hyperbolic geometry turned to the implementation of cellular automata in this context. A short introduction can also be found in [2]. However, it is not required to be an expert in hyperbolic geometry in order to read this paper.

Section 2 reminds the definition we take for weak universality. Section 3 is devoted to the proof of Theorem 2. In that section, Subsection 3.1 reminds the basic model used in the paper, Subsection 3.2 explains its implementation in the pentagrid, the tiling $\{5, 4\}$ of the hyperbolic plane, Subsection 3.2 explains the scenario of the simulation performed by the automaton proving Theorem 2.

2 Universality and Weak Universality

Universality is a well know notion in computer science. However, the single word 'universality' is understood in different ways, sometimes somehow divergent.

Let us go back to the definition.

C.S. Calude et al. (Eds.): Gruska Festschrift, LNCS 8808, pp. 99–113, 2014.
DOI: 10.1007/978-3-319-13350-8_8

Definition 1. *Let $\mathcal{K}$ be a class of processes. Say that $\mathcal{K}$* **possesses a universal element** *U if, a finite alphabet A being fixed once and for all, there is an encoding c of $\mathcal{K}$ elements into the words on A and of the data for a $\mathcal{K}$-element into the words on A such that for all element χ of $\mathcal{K}$ and for all data d of χ, U applied to $(c(\chi), c(d))$ ends its computation if and only if χ ends its own one when it is applied to d and, in that case, if $U(c(\chi), c(d)) = c(\chi(d))$.*

We also say that U **simulates** χ or that χ is **simulated** by U. If $\mathcal{K}$ possesses a universal element, we also say that $\mathcal{K}$ possesses the property of being universal. Note that we get again the standard definition of a universal Turing machine.

In the above definition, there are four elements. Data χ and d, the encoding of c and the universal element U. From the definition itself, the notion of encoding is an essential feature. Indeed, among the elements of $\mathcal{K}$, it is not difficult to construct some of them whose encoding is bigger than that of U. Indeed, it may be assumed that the encoding is an increasing function in this sense that if χ and ξ are elements of $\mathcal{K}$ transforming words on A onto words on A, then $c(\xi_\circ\chi) > c(\chi), c(\xi)$. Consequently, encoding the elements of $\mathcal{K}$ into a fixed alphabet is an essential feature. It allows U to simulate objects which are bigger than itself. Of course, changing the encoding may result in a change on the computation of U which may then be either faster or slower. At last, when U stops its computation, the result is an encoding of the result of the element of $\mathcal{K}$ simulated by U.

Now, since a few decades, these three items: the data, the encoding and the result are not always considered in the same way. From the definition, d is finite, as a word on A; when the computation of χ on d stops, that of U on $c(\chi)$ and $c(d)$ also stops. Now, whether this latter condition is observed or not, it happens that when U is applied to $c(\chi)$ and $c(d)$, it does not yield $c(d)$ when χ completes its computation on d, but something else, call it $e(d)$, where e can be considered as another encoding of d, e being also fixed once and for all. Indeed, $U(c(\chi), c(d)) = c(\chi(d))$ can also be rewritten $c^{-1}(U(c(\chi), c(d))) = \chi(d)$, so that $\chi(d)$ is restored by **decoding** $U(c(\chi), c(d)))$. Introducing e consists in accepting that the decoding function can be independent from the encoding one.

Now, the conditions of finiteness on d and on the computation of U when that of χ on the considered data stops are not always observed. When all conditions of Definition 1. are observed we say that U is **strongly universal**. In this definition of strong universality, it is not required that the decoding be the inverse function of the encoding. However, it is required that e and c belong to comparable classes of complexity. Specifically, it is required that there is a primitive recursive function u such that for any d, $c(d), e(d) \leq u(|d|)$, where $|d|$ is the size of $c(d)$, *i.e.* the number of symbols in $c(d)$.

When the conditions of strong universality are not observed, we say that U is **weakly universal**. In case d is in some sense infinite, it is required that $c(d)$, which is an infinite word be of the form u^*wv^*, where u, v and w are words on A. The repeated words u and v are called the periodic patterns and it is not required that $u = v$. Note that during the computation, we may consider that each step t of U works on something of the form $u^*w_tv^*$. It is this situation that we shall

consider, with this difference that we work in a $2D$-space and so, accordingly, we require a condition of periodicity restricted to the outside of a big enough disc, this condition being able to involve two different periodic patterns.

3 The Scenario of Our Simulation

Most of the cellular automata in hyperbolic spaces I constructed, myself or with a co-author, apply the same model of computation. We implement a railway circuit devised by Ian Stewart, see [8] which we rework in order to simulate a register machine. This general scenario is described in detail in [4,5]. Here, we simply give the guidelines in order to introduce the changes which are specific to this implementation.

3.1 Basic Features

The railway circuits consists of tracks, crossings and switches, and a single locomotive runs over the circuit. The tracks are pieces of straight line or arcs of a circle. In the hyperbolic context, we shall replace these features by assuming that the tracks travel either on **verticals** or **horizontals** and we shall make it clear a bit later what we call by these words. The crossing is an intersection of two tracks, and the locomotive which arrives at an intersection by following a track goes on by the track which naturally continues the track through which it arrived. Again, later we shall make it clear what this natural continuation is. Below, Figure 1 illustrates the switches and Figure 2 illustrates the use of the switches in order to implement a memory element which exactly contains one bit of information.

The three kinds of switches are the **fixed switch**, the **flip-flop** and the **memory switch**. In order to understand how the switches work, notice that in all cases, three tracks abut the same point, the **centre** of the switch. On one side switch, there is one track, say a, and on the other side, there are two tracks, call them b and c. When the locomotive arrives through a, we say that it is an **active** crossing of the switch. When it arrives either through b or c, we say that it is a **passive** crossing.

In the fixed switch, in an active passage, the locomotive is sent either always to b or always to c, we say that the **selected track** is always b or it is always c. In the passive crossing, the switch does nothing, the locomotive leaves the switch through a. In the flip-flop, passive crossings are prohibited: the circuit must be managed in such a way that a passive crossing never occurs at any flip-flop. During an active passage, the selected track is changed just after the passage of the locomotive: if it was b, c before the crossing, it becomes c, b respectively after it. In the memory switch, both active and passive crossing are allowed. The selected tracks also may change and the change is dictated by the following rule: after the first crossing, only in case it is active, the selected track is always defined as the track taken by the locomotive during its last passive crossing of the switch. The selected track at a given switch defines its **position**.

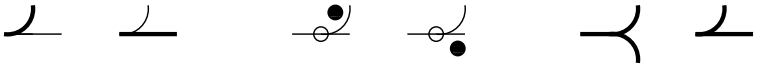

Fig. 1. The switches of the railway circuit. From left to right: the fixed switch, the flip-flop and the memory switch.

The current configuration of the circuit is the position of all the switches of the circuit. Note that it may be coded in a finite word, even if the circuit is infinite, as at each time, only finitely many switches have been visited by the locomotive.

Figure 2 illustrates how a flip-flop and a memory switch can be coupled in order to make a one bit memory element.

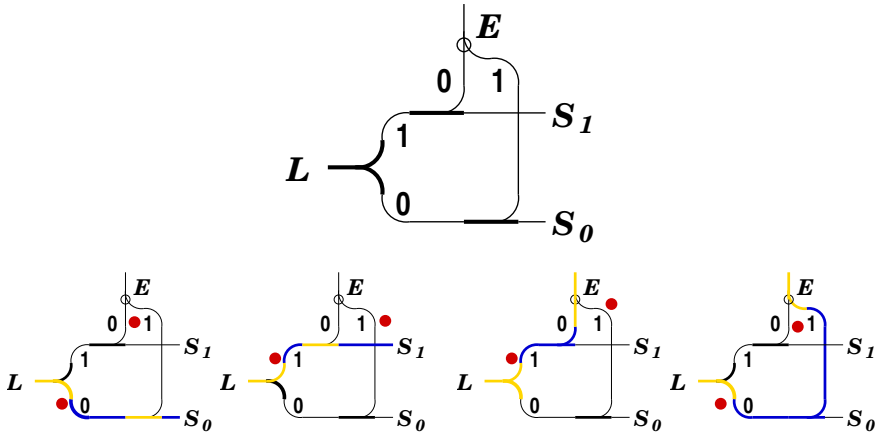

Fig. 2. The basic element of the circuit. Second row: Firs two drawings: reading the element. Last two drawings: writing the element.

3.2 In the Pentagrid

As mentionned in the introduction, the first weakly universal cellular automaton on the pentagrid was done by the author and a co-author, see [1]. The paper implements the solution sketchily mentioned in Subsection 3.1 with 22 states. In the next paper about a weakly universal cellular automaton on the pentagrid, see [7], the same model is implemented with 9 states. The difference with the former paper is that in the second paper, the cell which is at the centre of the switch has the same colour as another cell of the track. The centre of the switch is signalized by the neighbouring of the centre.

Former Implementations. Figure 4 illustrates the implementation of the crossing and of the switches performed in [7], showing in particular, the feature

at which we just pointed. Figure 3 shows the implementation of the verticals, second row in the figure, and of the horizontal, first row. Both these figures show how to implement the basic element of Figure 2 in Subsection 3.1. Figure 5 show a global view of how the tree structure of the tiling can be used to implement a basic element in the pentagrid.

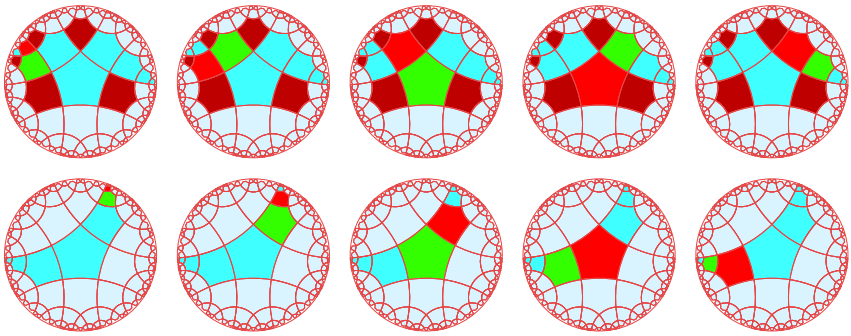

Fig. 3. Implementation of the tracks in the pentagrid. On the top, running a horizontal track from left to right. On the bottom, running a vertical track from top to bottom.

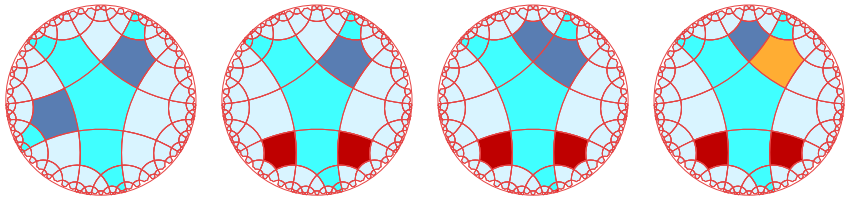

Fig. 4. Implementing the crossing and the switches in the pentagrid. From left to right; crossing, fixed switch, memory switch and flip-flop.

In this implementation, as well as in that of [1], the locomotive is implemented as two contiguous cell with different colours, which allows the implemented vehicle to find the direction of its motion along the tracks. The colours were chosen as green and red, green pointing at the front and red at the rear. The direction is then obvious. These are the elements which allowed us to prove the following result.

Theorem 1. (Margenstern, Song), *cf.*[7] – *There is a planar cellular automaton on the pentagrid with* 9*-states which is weakly universal and rotation invariant.*

By planar, we mean that the trajectory of the cells of the cellular automaton which at some point change their state is a planar structure which contains infinitely many cycles which cannot be reduced to a $1D$-structure. This is in particular the case of the units which constitute the registers of the register machine implemented by the railway circuit.

The New Scenario. In this paper, we take benefit of various improvements which I brought in the construction of weakly universal cellular automaton constructed in other contexts: in the heptagrid, another grid of the hyperbolic plane, in the hyperbolic $3D$-space and in the tiling $\{13, 3\}$ of the hyperbolic plane, that latter automaton having two states only, see [5] for details.

Our implementation follows the same general simulation as the one described in Subsection 3.1. In particular, Figure 5 is still meaningful in this new setting.

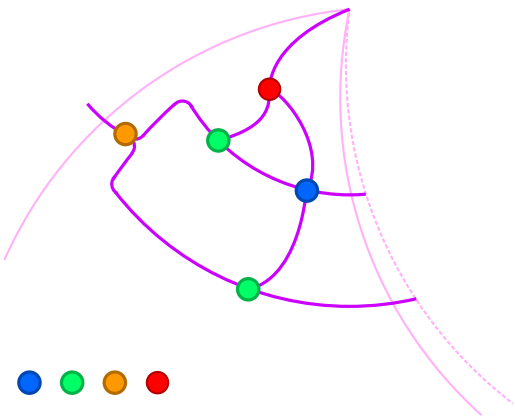

Fig. 5. Implementing the basic ellment in the petagrid. The discs on the bottom row, from left to right, represent the crossing, the fixed switch, the memory switch and the flip-flop.

However, here, new features are introduced.

The first change is that the tracks are one-way. In some sense this is closer to what we can see for railways in real life, in particular for highspeed ones. This change entails a big change in the switches and in the crossings. There is no change for the flip-flop which was already a one-way structure from the very beginning as passive crossings are ruled out for this kind of switches. For fixed switches it introduces a very small change: we keep the structure for a passive crossing and for the active one, as the selected track is the same, it is enough to continue the active way without branching at the centre of the switch, see Figure 6. In the same picture, we can see that the situation is different for the memory switch. This time, as there are two possible crossings of the switch and as the selected track may change, we have two one-way switches: an active one and a passive one. At first glance, the active switch looks like a flip-flop and the passive switch looks like a one-way fixed one. However, due to the working of the memory switch, we could say that the active memory switch is passive while the passive memory switch is active. Indeed, during an active crossing, the selected track is not changed contrary to what happens in the case of the flip-flop. Now, during a passive crossing, the switch looks at which track is crossed: the selected or the non-selected one. If it is the non-selected one, then the selection is changed

and this change is also transferred to the active switch. Accordingly, there is a connection between the active and the passive one-way memory switches.

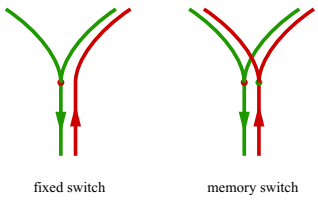

Fig. 6. The new switches for a one-way structured circuit: the fixed and the memory ones. Note that the flip-flop remains the same as in Figure 1.

Now, if we wish to significantly reduce the number of states, we also have to change the tracks themselves, as it appears that giving them the same colour as the background is better than assigning a special colour to identify the tracks. The consequence is that we have to place *milestones* in order to do so. This was performed in previous works, see [5]. But this is not enough: we also have to change the crossings. Contrary to what happens in the $3D$-space where crossings can be replaced by bridges, which makes the situation significantly easier, crossings cannot be avoided in the plane.

In [5] we indicate a solution which allowed me to build a weakly universal cellular automaton in the hyperbolic plane with 2 states only. However, this was not performed in the pentagrid nor in the heptagrid, but in the tiling $\{13, 3\}$. This solution can be implemented here and this allowed me to reduce the number of states from 9 down to 5. There is a slight improvement in the present solution which might allow us to reduce the number of neighbours for a two-state weakly universal cellular automaton in the hyperbolic plane.

First, we look at a crossing of two one-way tracks. The main idea is that we organize the crossing in view of a **round-about**: an interference of road trafic in our railway circuit. We may notice that the locomotive arriving either from **A** or **B** in Figure 7 has to turn right at the second pattern it meets on its way. So that it is enough to devise a pattern which allows to count from 1 up to 2 in some way. In the two-states world of the weakly universal cellular automaton on $\{13, 3\}$ described in [5], there were four patterns: a first one appears when the locomotive arrives at the round-about. At this point, a second locomotive is appended to the arriving one. At the second pattern, one locomotive is removed and so, as a single locomotive arrives through the round-about at the third pattern, then it knows that it has to turn right. Here, this scheme is slightly changed as follows. At the first pattern it meets the locomotive, which, arriving in a state `S`, is changed into `T`. At the second pattern, as a locomotive in the state `T` arrives, it is sent on the track which leaves the round-about. This is why three patterns are needed in Figure 7.

Figure 8 shows us how to assemble four one-way round-abouts in order to perform a true crossing for the intersection of two two-ways tracks.

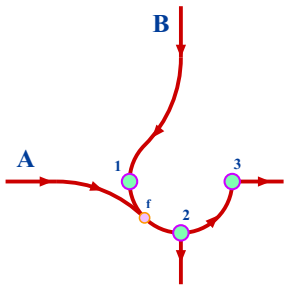

Fig. 7. The new crossing: the one-way tracks from**A** and **B** intersect. We have a three-quarters round-about. The small disc at **f** represents a fixed switch. Discs **1**, **2** and **3** represent the pattern which dispatches the motion of the locomotive on the appropriate way. Patterns **1** and **3** are needed as explained in the description of the scenario.

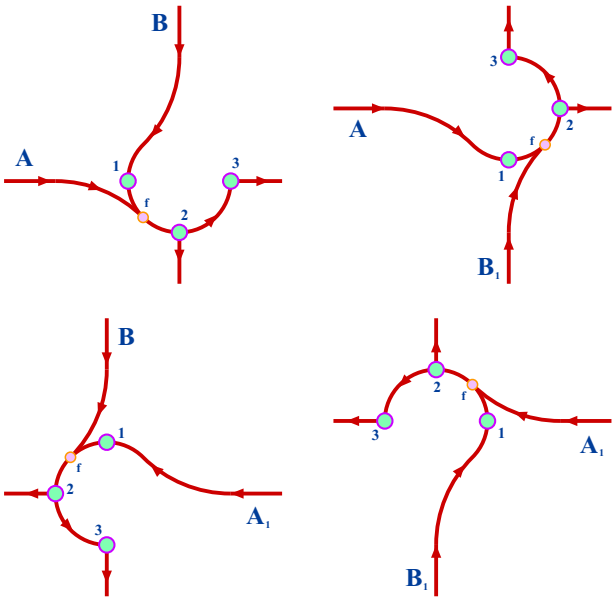

Fig. 8. The new crossing: four possible one-way track. Assembling them allows us to perform a two-way crossing. The notations are those of Figure 7. $\mathbf{A}_1$, $\mathbf{B}_1$ go opposite to **A**, **B**, respectively.

Now that we have seen the scenario to implement the circuit, we have to precisely look at how to implement it with five states only. We turn now to this question.

Implementing the New Scenario. As already announced, we need to use five states only. The first state is the quiescent state which we denote by `W`. Remember it is defined by the following rule: if a cell c and all its neighbours are in the state `W`, then at the next top of the clock, the cell c remains in the state `W`. We shall also call `W` the blank and we also shall say that a cell in `W` is white. The other states are `B`, `G`, `R` and `Y`. The cells in these states are said to be blue, green, red or yellow, respectively. The state `B` is mainly used for the milestones which delimit the tracks. The state `G` is the basic state of the locomotive. In crossings, the locomotive turns to `R`. The state `Y` is used for special markings in the passive memory switch. The states `B`, `G` and `R` are also used for marking in the crossings and in the other switches. The state `G` also appears in the milestones for the tracks.

Checking the new scenario requires to first study the implementation of the tracks. We have to define verticals and horizontals.

Vertical and horizontal tracks

The verticals are easy to define, they correspond to branches of the Fibonacci tree. In fact they are finite sequences of cells for which a side lies on a fixed line. The line can be changed inside the sequence as illustrated by Figure 9 for a green locomotive running on a top-down track.

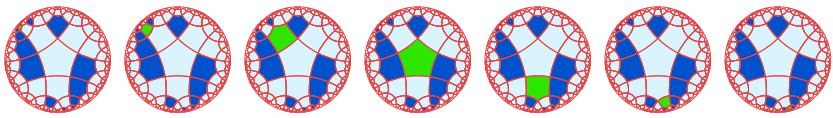

Fig. 9. Top-down traversal of a vertical with a green locomotive

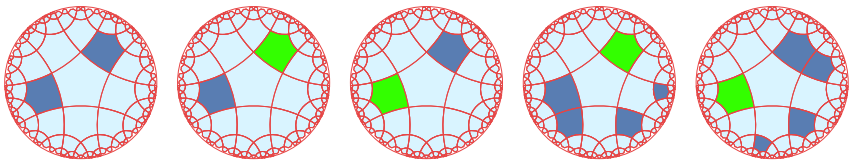

Fig. 10. The elements of the tracks. Leftmost picture: the standard element. Second and third pictures from left: the element which allows to perform sharp turns. Fourth and fifth pictures, illustration for a sharp turn.

The structure of an element of the track is simple. Assume that the locomotive leaves the cell through its side 1. Then, the milestones are always on sides 2 and 5. It enters either through sides 3 or 4. Both cases occur as illustrated in

Figure 9. Now, if we consider the standard numbering of the sides, then the place of the milestones depends on the direction of the motion. Assume that in the central cell, side 3 is the horizontal side. Then, for the cell 1(1), cell 1 of sector 1, the milestones are on the sides 3 and 5 in a bottom-up motion while they are on the sides 2 and 5 in the top-down one. The leftmost picture of Figure 10 illustrates the milestones as just defined.

The other pictures of Figure 10 illustrate two other patterns for the tracks. The second and third picture of the figure illustrates an element of the track which allows the locomotive, either green or red, to perform a sharp turn: this means that the locomotive enters through a side and exits through a contigous one. Such a turn is absolutely needed in the pentagrid in order to have cycles in the trajectory of the locomotive. If such a possibility would not be allowed, the locomotive would run to infinity without returning to any tile it had already visited.

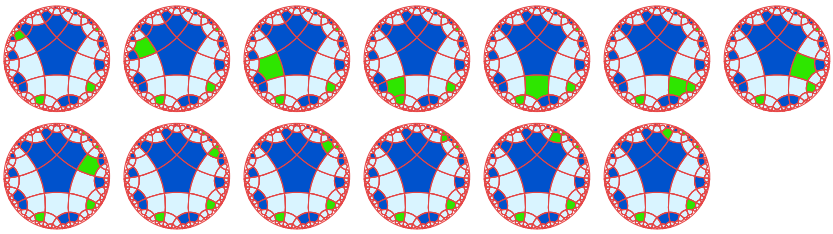

Fig. 11. A horizontal with a green locomotive, left-right run

The last two patterns of Figure 10 illustrate how to perform a sharp turn. A blue milestone is replaced by a green one. There are two possiblities and each of them is used for a direction of the motion. Rules are devised in such a way that the green milestone prevents a backward motion of the locomotive as can be checked on the pictures from the above descriptions.

It is the place to remark that as horizontal tracks run on three consecutive levels, a two-way portion of the tracks require at least seven consecutive levels as we need at least one level to separate the tracks run on each direction. A similar remark also holds for vertical lines. A two-way section must be separated by several nodes on the same level at the level where the distance between the supporting line is minimal: this distance must be positive, which guarantees that the lines are not secant. These constraints require much space for the implementation, but in the hyperbolic plane, we are never short of space.

Implementing the crossings

As indicated in Subsubsection 3.2, the tracks are organized according to what is depicted in Figures 7 and 8. From the latter figure, it is enough to focus on the implementation of Figure 7. From the implementation of the tracks, we only have to look at the implementatation of the patterns symbolically denoted as **1**, **2**, **3** and **f** in the figure. As **f** is a fixed switch whose implementation is indicated a bit further, we simply implement **1** as the other patterns are strict copies of this one. Figure 12 illustrates this implementation and the behaviour of the

locomotive when it crosses the pattern. We can see that the difference strongly depends on the colour of the locomotive.

From Figure 7, the locomotive always arrives to the pattern from the same cell, namely 4(4). Then the locomotive goes to the centre, cell 0.

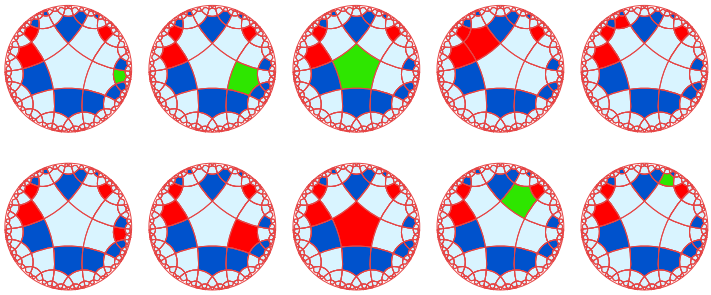

Fig. 12. The key pattern of the crossings. Notice the difference of behaviour depending on the colour of the locomotive.

When the locomotive is green, it goes to cell 1(1) arriving there as a red cell, and then it goes out onto the round-about towards the next pattern, still as a red locomotive. When the locomotive which arrives at cell 4(4) is red, it also goes to 0 but from there, it goes to 1(5) where it arrives as a green cell. Indeed, cell 1(5) remains white until it sees a red cell through its side 1. Note that side 1 is clearly indentified thanks to the pattern of the neighbours of cell 1(5). It is the pattern of an element of the tracks with a red milestone in place of the green one.

Implementation of a fixed switch

Figure 13 illustrates the passive crossing of a fixed switch for a green locomotive.

With one-way tracks, we need a single kind of passive fixed switch as there is no need of an active fixed switch, see Figure 6.

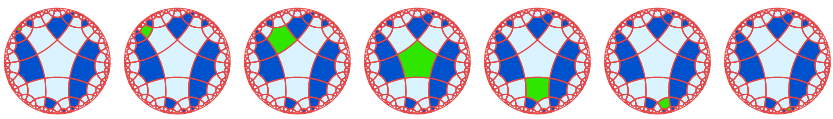

Fig. 13. The fixed switch: passive crossing by the green locomotive

Implementation of a flip-flop

With the flip-flop, we introduce the fifth state, `Y`. Figure 14 illustrates the configuration of a flip-flop and its active crossing by a locomotive.

Note that the change of the selected track occurs some time after the locomotive left the switch. The white cell at 2(2), it has three red neighbours, detects the passage of the locomotive just before the latter leaves the switch. Then, it

passes the information to the cell 2(1) through 1(1). This makes 2(1) to flash, turning from Y to R and then turning back to Y. This flash makes both 1(1) and 1(5) to change their states in a way which triggers 2(2) and 2(5) to change their states. When the cell 2(2) is red, a symmetric process occurs.

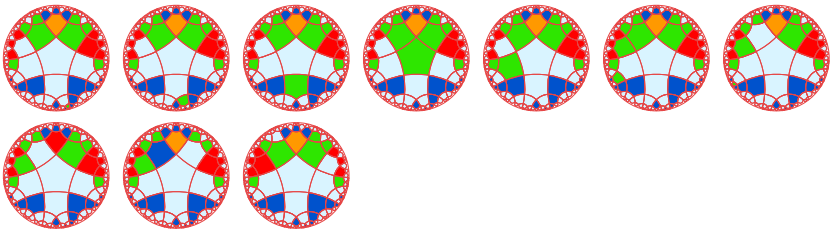

Fig. 14. The flip-flop

Implementation of a memory switch

Now, we arrive to the most difficult situation. We have to implement two switches with a connection between them. As already noticed in Subsection 3.2, the active switch has a passive behaviour when crossed by the locomotive and the passive switch has an active behaviour when the locomotive takes the non-selected track. This action of the passive switch triggers the change of selection in the active switch: hence we have to organize the connection from the passive switch to the active one. The pattern of these switches, when the locomotive is not present, is illustrated by Figure 15.

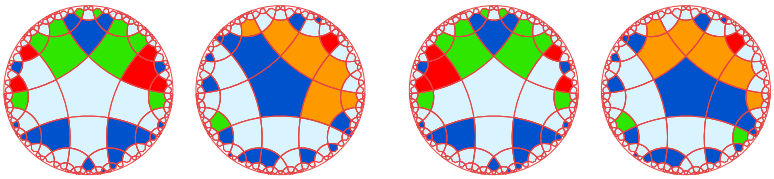

Fig. 15. The stable configuration of the active and passive memory switches. To left, the switches selected the right-hand side track. To right, they selected the left-hand side track.

Figure 16 illustrates the crossing of the active switch by the locomotive. Due to its way of working, the memory switch has two basic positions according to which is the selected track. We say that the left-, right-hand side switch selects the left-, right-hand side track respectively. We can check on the figure that the switch remains unchanged after the traversal of the locomotive.

We can see that the pattern of the active memory switch looks like that of the flip-flop. The difference is restricted to the cell 2(1) and its neighbours. In the flip-flop, the cell 2(1) is in Y and three consecutive neighbours are in B: cells 5, 6 and 7 of sector 1. In the active memory switch, the cell 2(1) is in B and the

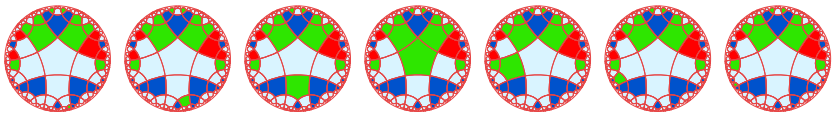

Fig. 16. The active memory switch. The left-hand side switch.

cell 6(1) is white. Moreover, the cells 15(1) and 18(1) are in `G`. We shall see the role of these green cells in a while.

Before, we look at the crossing of the passive memory switch. We have four situations as the switch has two positions and as for each position, the locomotive may arrive either through the selected track or through the non-selected one.

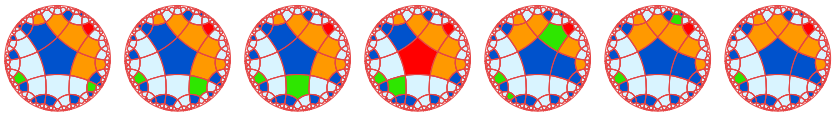

Fig. 17. The passive memory switch. Here a left-hand side switch and a passive crossing through the non-selected track. Note that the selected track is changed in the right-hand side half of the figure.

As is clear on Figure 15, the pattern of the passive memory switch is very different from the active one. Although a part of it is taken from a fixed switch whose centre would be at the cell 2(3), the surrounding of this cell and the passive tracks is very specific.

Figure 17 illustrates the case when the locomotive crosses the non-selected track. We can see that this changes the selection according to the definition of the memory switch: the non-selected track becomes the new selected track.

Now, this information has to be transferred to the active memory switch. This is performed by the pattern of the passive memory switch. The crossing through the non-selected track is detected by the cell 1(5) which is in contact with the central cell. That latter one has a `B`-neighbour on the side of the selected track and a `Y`-one on the side of the non-selected track. When the locomotive becomes a neighbour of the central cell, it abuts the cell on the side of the `Y`-neighbour or of the `B`-one. This allows the central cell to know through which track the locomotive has run. Accordingly, if the run went through the non-selected one, the central cell flashes: it turns from `B` to `R` and then turns back to `B`. Now, this flash makes the cells 1(1) and 1(4) to take the opposite colour, from `B` to `Y` or from `Y` to `B`: this changes the signalization of the non-selected track. But the cell 1(5) also can see the flash of the central cell. This makes the cell 1(5) to also flash: it turns to `G` and then turns back to `Y`. Now, the cells 5(1) and 11(5) are milestones for the cell 4(5) which, accordingly, appears to be a possible element of a track. Consequently, the flash of 1(5) creates a second locomotive which can

go along the track whose starting point is cell 4(5). It is enough to define a track going to the active memory switch to make the needed connection between the two parts of the memory switch.

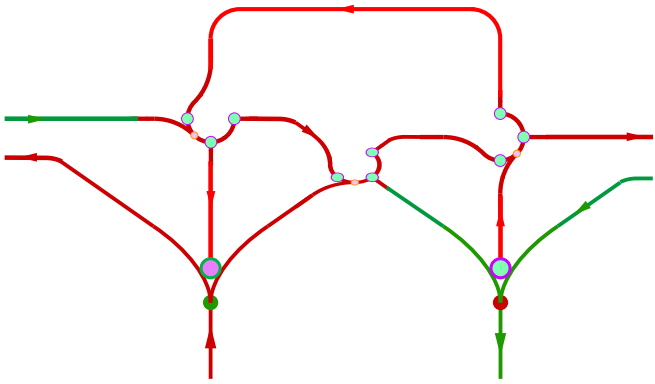

Fig. 18. The organisation of the memory switch

Figure 18 illustrates the global setting for implementing the memory switch with its tow parts, the passive and the active one and the connection between them. Now, the path whose starting point is the cell 4(5) in the passive switch, see Figure 15, goes to the active switch as indicated in Figure 18 and it arrives at the cell 6(1) of the active switch, see Figure 15.

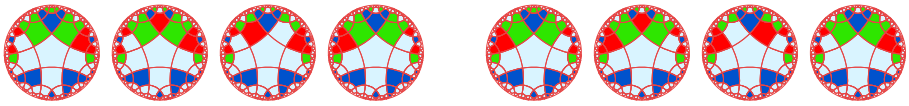

Fig. 19. The change of selection in the active memory switch triggered by the arrival of the second locomotive at the cell 6(1). To left the case when the left-hand side track is selected, to right, when it is the case for the right-hand side track. The presence of the locomotive in 6(1) can be seen in the first picture of each series.

The cells 15(1) and 17(1) allow to make the second locomotive sent from the passive switch go to the cell 16(1) from where it is driven to the cell 6(1). Now, when the cell 2(1) can see the second locomotive, it flashes, turning to `R` and then back to `B`, which makes the cells 1(1) and 1(5) trigger the signal to the cells 2(2) and 2(5) which then take the opposite colour.

Figure 19 illustrates the situation when the second locomotive arriving at the cell 6(1) triggers the change of selection. As can be seen in the figure, the second locmotive vanishes just after it arrived at 6(1). This arrival makes the cell 2(1) flash and then the same mechanism as seen for the flip-flop apply: the situation for cells 2(2) and 2(5) is exactly the same.

And so, outside the locomotive which yields the simulation of the computation in this model, call it the **main locomotive**, from time to time a second locomotive appears for a while in order to transmit the appropriate signal to an active memory switch. It is important to notice that the motion of this second locomotive does not interfer with the motion of the main one. Indeed, although the track from a passive memory cell to its corresponding active one is very long, the distance between whole switches is much larger. In any case it can be made much larger: this can easily be seen on Figure 5. Also note that the second locomotive may sometimes be red. Indeed, as indicated by Figure 18, the second locomotive travels through two crossings.

With this study illustrated by the figures and the rules which can be found in [6], we completed the proof of the following result:

Theorem 2. *There is a rotation invariant cellular automaton on the pentagrid with* 5 *states which is planar and weakly universal.*

References

1. Herrmann, F., Margenstern, M.: A universal cellular automaton in the hyperbolic plane. Theoretical Computer Science **296**, 327–364 (2003)
2. Margenstern, M.: Cellular Automata and Combinatoric Tilings in Hyperbolic Spaces, a survey. In: Calude, C., Dinneen, M.J., Vajnovszki, V. (eds.): DMTCS 2003. LNCS, vol. 2731, pp. 48–72. Springer, Heidelberg (2003)
3. Margenstern, M.: Cellular Automata in Hyperbolic Spaces. In: Adamatzky, A. (ed.) Theory, Collection: Advances in Unconventional Computing and Cellular Automata, vol. 1, p. 422. Old City Publishing, Philadelphia (2007)
4. Margenstern, M.: Cellular Automata in Hyperbolic Spaces. In: Adamatzky, A. (ed.) Implementation and computations. Collection: Advances in Unconventional Computing and Cellular Automata, vol. 2, p. 360. Old City Publishing, Philadelphia (2008)
5. Margenstern, M.: Small Universal Cellular Automata in Hyperbolic Spaces: A Collection of Jewels, Collection: Emergence, Complexity and Computation. In: Zelinka, I., Adamtzky, A., Chen, G. (eds.) p. 331. Springer (2013), doi:10.1007/978-3-642-36663-5
6. Margenstern, M.: A weakly universal cellular automaton in the pentagrid with five states, p. 23. arXiv:1403.2373
7. Margenstern, M., Song, Y.: A new universal cellular automaton on the pentagrid. Parallel Processing Letters **19**(2), 227–246 (2009). doi:10.1142/S0129626409000195
8. Stewart, I.: A Subway Named Turing, Mathematical Recreations in Scientific American, 90–92 (1994)