Synchronizing a virtual cellular automaton collider

Oswaldo L. Barrientos *, Genaro J. Martínez * †, Andrew Adamatzky †, Shigeru Ninagawa ‡

* *Computer Science Laboratory, Escuela Superior de Cómputo, Instituto Politécnico Nacional, México*
† *Unconventional Computing Laboratory, University of the West of England, Bristol, United Kingdom*
‡ *Kanazawa Institute of Technology, Hakusan, Japan*

2022 Tenth International Symposium on Computing and Networking (CANDAR) | 978-1-6654-7530-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CANDAR57322.2022.00009

***Abstract*—Cellular automata have been a topic of great interest in unconventional computation theory for its architecture: simple but capable of producing complex structures with a dynamic behavior and massive parallel computation. Collision-based computing with multiple reactions is a way to explore such capacities. Following this logic, a virtual collider was proposed as a way to develop computers by using the elementary cellular automaton rule 110. The main result in this paper is the computational implementation of the virtual collider, capable of supporting the execution of three simultaneous cyclotrons. The lateral cyclotrons handle particles which are injected to the main collider to perform a computation. Particles move until they reach a contact point, where they trigger a series of chained reactions among hundreds of thousands of cells. Using this program, we present a configuration of the collider that simulates the execution of the first nine transitions of a cyclic tag system using rule 110 particles, the largest simulation of this kind to date.**

***Index Terms*—cellular automata, rule 110, cyclic tag system, virtual collider, unconventional computing.**

I. Introduction and basic notation

Cellular automata

The study of the dynamic operations of cellular automata (CA) have become a significant area of mathematics and theoretical computation in the last years. CA provide a framework for a class of dynamic discrete systems that give rise to a complex and unpredictable behavior, emerging from local and deterministic interactions between components acting in parallel [1].

A CA is defined by the 4-tuple:

$$M \coloneqq (d, \Sigma, N, \phi)$$

Where:

- $d \in \mathbb{Z}^+$ is the dimension of M. The cellular space of M is $\mathbb{Z}^d$, and all the vectors $c \in \mathbb{Z}^d$ are called cells.
- Σ is an alphabet of states of cardinality k, commonly represented with the integers $\{0, 1, \ldots, k-1\}$. The configuration of M is a function $s : \mathbb{Z}^d \to \Sigma$ that assigns a state to each cell.
- $N : \mathbb{Z}^d \to (\mathbb{Z}^d)^n$, with $n \in \mathbb{Z}^+$, is a neighborhood function for M. For any cell $c \in \mathbb{Z}^d$, $N(c)$ outputs the cells $c_1, c_2, \ldots, c_n \in \mathbb{Z}^d$ that are part of the neighborhood of c.
- $\phi : \Sigma^n \to \Sigma$ is a function, called transition rule, that determines the new state of a cell from the states of the cells in its neighborhood. For any cell $c \in \mathbb{Z}^d$ with neighborhood $N(c) = (c_1, c_2, \ldots, c_n)$ at time t_i, $\phi(s(c_1), s(c_2), \ldots, s(c_n))$ produces a new state for c at time t_{i+1}. This rule is applied synchronously among all cells in M, producing a new global state for each step of time [2].

An elementary cellular automaton (ECA) is a one-dimensional ($d = 1$) CA that uses binary values for its cells ($\Sigma = \{0, 1\}$) and a 3-cell local neighborhood ($n = 3$ and $N(i) = (i-1, i, i+1)$ for $i \in \mathbb{Z}$). With these characteristics, there are 8 possible states for the neighborhood, and thus 2^8 different rules. If the outputs of an elementary rule are ordered by the binary value given by its input, it is possible to code the resulting states to give the rule a unique identifier. For example, output states $0, 1, 1, 0, 1, 1, 1, 0$, define the rule $(01101110)_b$, also called ECA rule 110.

The evolution space for the first iterations of this rule using a random initial configuration for the cellular space is shown in figure 1. For practical purposes, the cellular space of the CA is considered large enough, and its border cells are processed in a periodic manner.

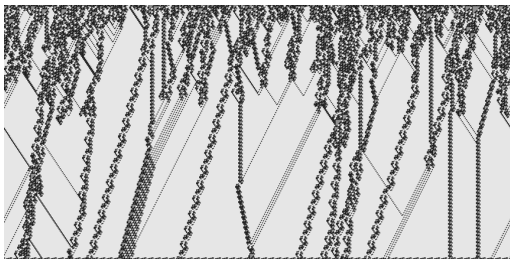


Fig. 1: Classic evolution of ECA rule 110 from a random initial configuration at 50%, evolving on a space of 1000 cells through 500 generations. A filter is applied on the periodic background for a better visualization of particle dynamics.

Rule 110

ECA rule 110 was proved to be capable of universal computation by Cook [3]. He analyzed the evolution space of rule 110, isolating the binary strings that were able to create defined patterns, which he called particles. These particles seemed to move through a periodic background, informally named *ether*. The set of all the particles of rule 110 identified by Cook is defined by:

$$\begin{aligned}\Gamma \coloneqq \{&e, A, A^2, A^3, \ldots, B, \bar{B}, \bar{B}^2, \bar{B}^3, \ldots, \hat{B}, \hat{B}^2, \hat{B}^3, \ldots,\\ &C_1, C_2, C_3, D_1, D_2, E, E^2, E^3, \ldots, \bar{E}, F,\\ &G, G^2, G^3, \ldots, H, gun, gun^2, gun^3, \ldots\}\end{aligned}$$

2379-1896/22/$31.00 ©2022 IEEE
DOI 10.1109/CANDAR57322.2022.00009

Figure 2 shows some of the particles of rule 110, isolated for better visualization of their shape and movement.

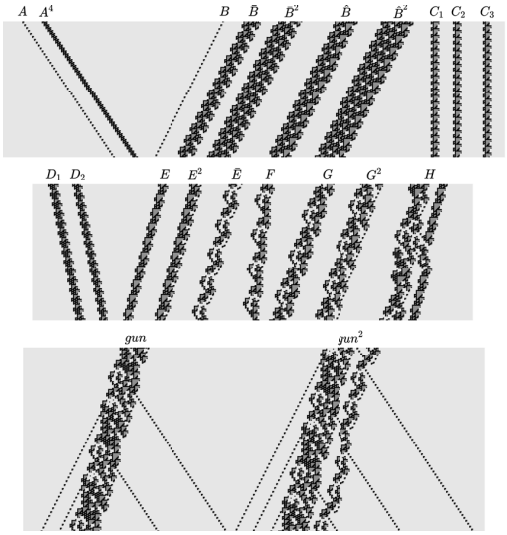


Fig. 2: Particles of rule 110.

For any two particles $\alpha, \beta \in \Gamma$, the notation $\alpha - \beta$ describes a concatenation of α with β. Furthermore, for particle $\alpha \in \Gamma$ and $a \in \mathbb{Z}^+$, we say that $a\alpha$ is a concatenation of particle α with itself a times [4].

Any particle $\alpha \in \Gamma$ has a list $\varphi_0, \varphi_1, \ldots, \varphi_{T-1} \in \{0,1\}^*$ of unique representations, called phases of α. $T \in \mathbb{Z}^+$ is called the period of α. When applying rule 110 to a phase φ_i, with $i \in \{0, \ldots, T-1\}$, the resulting space will contain $\varphi_{i+1 \bmod T}$, typically shifted a number of cells depending on α's specific horizontal displacement.

We introduce the notation for the phase of a particle as:

$$\alpha(\omega, f_i\_1)$$

Where:

- $\alpha \in \Gamma$ is a particle with period T and phases $\varphi_0, \varphi_1, \ldots, \varphi_{T-1}$.
- ω is an entry of the finite ordered list $\Omega = (A_1, B_1, C_1, \ldots, H_1, A_2, B_2, C_2, \ldots, H_2, \ldots)$. Ω represents sections of 4 phases of α, thus $|\Omega| = \lceil \frac{T}{4} \rceil$. For $i \in \{0, \ldots, |\Omega| - 1\}$, $\Omega(i)$ denotes the section of 4 phases φ_{4i}, φ_{4i+1}, φ_{4i+2} and φ_{4i+3} of α, if they exist. Entries $A_1, B_1, C_1, \ldots, H_1$ of Ω are commonly written as $A, B, C, \ldots, H$, respectively, or omitted altogether if $|\Omega| \leq 8$.
- f_i_1, with $i \in \{1, 2, 3, 4\}$, is the ith phase of α at section ω, given it exists.

In this manner, ether (e) has 4 different phases, which are $e(f_1\_1) = 11111000100110$, $e(f_2\_1) = 10001001101111$, $e(f_3\_1) = 10011011111000$ and $e(f_4\_1) = 10111110001001$.

J. Martínez et al. presented a compilation of the phases for the particles in rule 110 in [5], as well as a thorough analysis on the behavior of each individual particle in [6].

From now on, we will refer to $e(f_1\_1)$ as simply e.

Cyclic tag systems

A cyclic tag system (CTS) is a computational model created by Cook to show that ECA rule 110 is universal. The CTS is itself a universal system, as it is capable of emulating m-tag systems, another kind of system that can in turn emulate Turing machines [3].

A CTS of cycle length $k \in \mathbb{Z}^+$ is defined by the k-tuple:

$$C \coloneqq (p_0, \ldots, p_{k-1})$$

Where $p_0, \ldots, p_{k-1} \in \{0,1\}^*$ are called the productions of C [7]. An instantaneous description (ID) of C is represented by the tuple (v, m), where $v \in \{0,1\}^*$ and $m \in \{0, \ldots, k-1\}$. We call m a phase of the ID.

A transition relation $\Rightarrow$ in the set of the IDs is defined in the following manner: Let $(v, m), (v', m') \in \{0,1\}^* \times \{0, \ldots, k-1\}$. Then:

$$\begin{aligned}(1v, m) \Rightarrow (v', m') \text{ iff } &(m' = m + 1 \bmod k) \\ &\wedge (v' = vp_m), \\ (0v, m) \Rightarrow (v', m') \text{ iff } &(m' = m + 1 \bmod k) \\ &\wedge (v' = v)\end{aligned}$$

A sequence of IDs $(v_0, m_0), \ldots, (v_n, m_n)$ of C is a computation beginning in $v \in \{0,1\}^*$ if and only if:

$$(v_0, m_0) = (v, 0) \wedge (v_i, m_i) \Rightarrow (v_{i+1}, m_{i+1}) \; \forall i \in \{0, \ldots, n-1\}$$

[8]

C is said to halt on tape v if its computation reaches the empty word ϵ in a finite amount of transitions.

Figure 3 shows the first 20 transitions of the computations of a CTS with productions $(1, 101)$, starting with the initial tape 1. The resulting computation is $(1, 0) \Rightarrow \ldots \Rightarrow (1101111011, 0)$. It proves difficult to predict if in any moment in the future the CTS will halt.

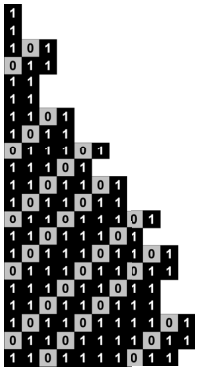
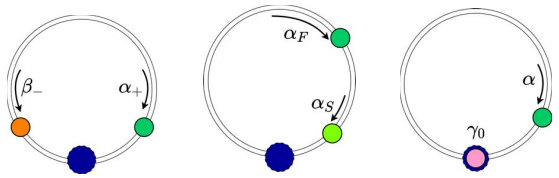


Fig. 3: First 20 transitions of the computation of a CTS with productions $(1, 101)$ starting with tape 1.

Symbolic collisions

In the 70s, Fredkin and Toffoli [9] proposed a model of computation based in ballistic collisions between quanta of information represented by abstract particles. Later, Margolus



(a) Two particles $\alpha+$ and β_- with opposite velocities move towards a collision point.

(b) A high velocity particle α_F moves towards a low velocity particle α_S.

(c) A particle α moves towards a stationary particle γ_0.

Fig. 4: Beam routings for different scenarios of collisions between two particles in an ECA.

[10] applied this concept to CA theory through the implementation of a neighborhood model that used ballistic collisions.

To represent a collision in an ECA, we use the notion of idealized particles (with no energy or potential), which move through a periodic background with either a positive, negative or neutral lateral displacement.

When assuming periodic boundary conditions, the cellular space of an ECA can be represented as a ring. Particles moving through this space can be transmitted through the boundary and interact indefinitely, resembling the operation of a virtual cyclotron.

In order to characterize the interactions in ECA rule 110, we will define a special notation for the collision of a two-particle system. Let $\alpha, \beta, \gamma, \gamma_1, \ldots, \gamma_m \in \Gamma$ and $m \in \mathbb{Z}^+$. The notation $(\alpha \Leftrightarrow \beta) \rightarrow (\gamma_1, \ldots, \gamma_m)$ indicates that α and β collide to create the m ordered particles $\gamma_1, \ldots, \gamma_m$. We will use ϵ to denote the absence of particles.

Collisions are classified as follows:

- Destruction: $(\alpha \Leftrightarrow \beta) \rightarrow \epsilon$.
- Fusion: $(\alpha \Leftrightarrow \beta) \rightarrow (\gamma)$.
- Interaction and production of new particles: $(\alpha \Leftrightarrow \beta) \rightarrow (\gamma_1, \ldots, \gamma_m)$.
- Identity or solitonic collision: $(\alpha \Leftrightarrow \beta) \rightarrow (\alpha, \beta)$.
- Reflection or elastic collision: $(\alpha \Leftrightarrow \beta) \rightarrow (\beta, \alpha)$.

The description of the dynamics for collisions in a cyclotron can be represented in an abstract diagram called a beam routing. Figures 4a, 4b and 4c show the beam routings for different scenarios of collisions between two particles towards a collision point in a cyclotron:

- Two particles, α_+ and β_-, move in opposite directions until they collide in the collision point (figure 4a).
- Two particles, α_F and α_S, move in the same direction, and α_F has a higher lateral displacement than α_S. The collision point marks the moment in which α_F closes up on α_S (figure 4b).
- A particle α moves towards a neutral particle γ_0. The collision point is found on top of γ_0 (figure 4c).

II. Computability in rule 110

If rule 110 is able to emulate a CTS, it follows that rule 110 is Turing-complete and thus universal. In this work, we will focus on simulating a CTS with productions $(1, 101)$ operating on an initial tape 1.

The particles of rule 110 are used to build composite structures that we will call particle packages. [4] and [11] provided a detailed characterization of these structures. Each one has its own functionality and interactions that will be useful for the simulation of the CTS.

We will briefly describe each of the particle packages as well as the construction for their first phases, applying the same phase notation used for the elementary particles.

Package 4_A^4

The package 4_A^4 defines four trains of A^4 particles, changing periodically in phase. The three phases for this package are used regularly, and thus are uniquely named $4\_A^4(F_1)$, $4\_A^4(F_2)$ and $4\_A^4(F_3)$.

$$\begin{aligned}
4_A^4(F_1) = {} & A^4(f_1_1) - 27e - A^4(f_3_1) - 23e - A^4(f_2_1) \\
& - 25e - A^4(f_1_1) \\
4_A^4(F_2) = {} & A^4(f_2_1) - 27e - A^4(f_1_1) - 23e - A^4(f_3_1) \\
& - 25e - A^4(f_2_1) \\
4_A^4(F_3) = {} & A^4(f_3_1) - 27e - A^4(f_2_1) - 23e - A^4(f_1_1) \\
& - 25e - A^4(f_3_1)
\end{aligned}$$

Packages $1Ele\_C_2$ *and* $0Ele\_C_2$

The package $1Ele\_C_2$ represents a 1 in the tape of the CTS, whereas $0Ele\_C_2$ represents a 0.

$$\begin{aligned}
1Ele_C_2(A, f_1_1) = {} & C_2(A, f_1_1) - 2e - C_2(A, f_1_1) - 2e \\
& - C_2(A, f_1_1) - e - C_2(B, f_2_1) \\
0Ele_C_2(A, f_1_1) = {} & C_2(A, f_1_1) - 2e - C_2(A, f_1_1) - e \\
& - C_2(A, f_4_1) - e - C_2(A, f_2_1)
\end{aligned}$$

Package $0Blo\_\bar{E}$.

The package $0Blo\_\bar{E}$ represents a 0 in a CTS production and generates the package $0Add\_\bar{E}$.

$$\begin{aligned}
0Blo_\bar{E}(A, f_1_1) = {} & \bar{E}(A, f_1_1) - 2e - \bar{E}(D, f_1_1) \\
& - \bar{E}(G, f_2_1) - \bar{E}(A, f_2_1) - 2e \\
& - \bar{E}(B, f_1_1) - 2e - \bar{E}(A, f_1_1) \\
& - e - \bar{E}(D, f_3_1) - 2e \\
& - \bar{E}(H, f_1_1) - \bar{E}(B, f_4_1) \\
& - \bar{E}(D, f_4_1) - e - \bar{E}(E, f_3_1) \\
& - e - \bar{E}(D, f_3_1)
\end{aligned}$$

Packages $1BloP\_\bar{E}$ *and* $1BloS\_\bar{E}$

Because of the asymmetric evolution of rule 110, it is necessary to use two different packages to represent a 1 in a CTS production: $1BloP\_\bar{E}$ (primary package) and $1BloS\_\bar{E}$ (standard package). Either package generates a package $1Add\_\bar{E}$.

$$
\begin{aligned}
1BloP\_\bar{E}(A, f_1\_1) = \bar{E}(A, f_1\_1) &- \bar{E}(B, f_3\_1) - e \\
&- \bar{E}(B, f_4\_1) - \bar{E}(D, f_4\_1) - e \\
&- \bar{E}(E, f_3\_1) - e - \bar{E}(D, f_3\_1) \\
&- \bar{E}(G, f_1\_1) - 2e - \bar{E}(B, f_1\_1) \\
&- \bar{E}(D, f_4\_1) - \bar{E}(G, f_2\_1) - 2e \\
&- \bar{E}(H, f_1\_1) - 2e - \bar{E}(G, f_1\_1) \\
1BloS\_\bar{E}(A, f_1\_1) = \bar{E}(A, f_1\_1) &- 2e - \bar{E}(D, f_1\_1) - e \\
&- \bar{E}(G, f_2\_1) - \bar{E}(A, f_2\_1) - 2e \\
&- \bar{E}(B, f_1\_1) - 2e - \bar{E}(A, f_1\_1) \\
&- \bar{E}(C, f_1\_1) - 2e - \bar{E}(F, f_2\_1) \\
&- \bar{E}(A, f_2\_1) - \bar{E}(C, f_2\_1) - 2e \\
&- \bar{E}(D, f_1\_1) - 2e - \bar{E}(C, f_1\_1)
\end{aligned}
$$

Package $SepInit\_E\bar{E}$

The package $SepInit\_E\bar{E}$ acts as the leader component for the interactions for the rest of the packages. It is essential to separate sectors of data and to determine the incorporation of data to the tape of the CTS by deciding whether the interactions of the sector will add or remove data.

$$
\begin{aligned}
SepInit\_E\bar{E}(A, f_1\_1) = E^5(A, f_1\_1) &- E^2(D, f_2\_1) - 3e \\
&- E^4(A, f_1\_1) - e - \bar{E}(G, f_2\_1) \\
&- e - \bar{E}(H, f_2\_1) - 2e \\
&- \bar{E}(A, f_2\_1) - \bar{E}(H, f_3\_1) - e \\
&- \bar{E}(G, f_3\_1)
\end{aligned}
$$

Packages $1Add\_\bar{E}$ *and* $0Add\_\bar{E}$

The package $1Add\_\bar{E}$ is generated by either $1BloP\_\bar{E}$ or $1BloS\_\bar{E}$, whereas $0Add\_\bar{E}$ is generated by $0Blo\_\bar{E}$.

$$
\begin{aligned}
1Add\_\bar{E}(A, f_1\_1) = \bar{E}(A, f_1\_1) &- 7e - \bar{E}(B, f_4\_1) \\
&- 5e - \bar{E}(C, f_1\_1) - 7e \\
&- \bar{E}(D, f_4\_1) \\
0Add\_\bar{E}(A, f_1\_1) = \bar{E}(A, f_1\_1) &- 7e - \bar{E}(B, f_4\_1) \\
&- 6e - \bar{E}(D, f_3\_1) - 6e \\
&- \bar{E}(E, f_3\_1)
\end{aligned}
$$

III. Description of the system

The complete system to simulate the CTS with productions $(1, 101)$ on an initial tape 1, can be divided in three main components:

- A left section, composed of trains of A^4 particles moving to the right, controls the writing of values added to the tape of the CTS.
- A central section, composed of a single particle $1Ele\_C_2$, contains the initial value 1 of the tape.
- A right section, composed of sectors of $\bar{E}$ particles moving to the left, is in charge of processing data, producing the necessary elements to add new values to the tape depending on the values found at the moment. A sector is composed of a leading package $SepInit\_E\bar{E}$, followed by a group of block packages $Blo\_\bar{E}$ that represent the production of the CTS at a specific transition. Therefore, the production list of the CTS must be coded as concatenated sectors in the right section. In order to simulate the cycling of the production list, this concatenation is carried out indefinitely. To activate the first data sector, an activator particle A^3 is placed at the beginning of the right section.

The simulation of the CTS process the tape from right to left in the following manner:

1) The $SepInit\_E\bar{E}$ that leads the first processing sector of the right section, collides with the activator particle A^3 to enter activation mode.
2) When activated, a $SepInit\_E\bar{E}$ collides with a static package from the central section, deleting it. This action simulates the deletion of the leading value from the tape of the CTS.
   a) If the deleted static package is a $0Ele\_C_2$, all the blocks $Blo\_\bar{E}$ in the sector will be destroyed. The destruction of a data sector represents the action of reading a value 0 from the tape, with which no new value is added.
   b) If the deleted static package is a $1Ele\_C_2$, the blocks $Blo\_\bar{E}$ to its right will collapse one by one to create addition packages $Add\_\bar{E}$.
      i) If the collapsing block is a $0Blo\_\bar{E}$, the resulting collision will create a $0Add\_\bar{E}$.
      ii) If the collapsing block is either a $1BloP\_\bar{E}$ or a $1BloS\_\bar{E}$, the resulting collision will create a $1Add\_\bar{E}$. If a sector starts with a value 1, a primary block must be used. For additional 1 values, the standard block is used.
3) As a result of the collisions between packages, individual $\bar{E}$ particles will be generated as residues. If the system is correctly synchronized in phase and distance, this residues will cross paths with other particles without altering their structure, effectively acting as solitons.
4) The addition packages generated in a sector will collide with the train of 4_A^4 particles from the left section, producing either a $0Ele\_C_2$ from a $0Add\_\bar{E}$, or a $1Ele\_C_2$ from a $1Add\_\bar{E}$. This collision represents the writing of a new value 0 or 1, respectively, at the end of the CTS tape.
5) Finally, after either collapsing or destroying an entire sector, a train of three A particles will be generated. This train will serve as an activator for the $SepInit\_E\bar{E}$ of the next sector.

In 2017, J. Martínez et al. [4] presented a system on a space of 56,240 cells, capable of simulating the CTS up to five transitions. This research will propose an optimization of the original system, reducing the the size of the cellular space and being able to process up to nine transitions of the CTS.

IV. Cyclotron beam routing and virtual collider

We can represent collisions in an ECA as transitions between beam routings, which is useful for designing a model for

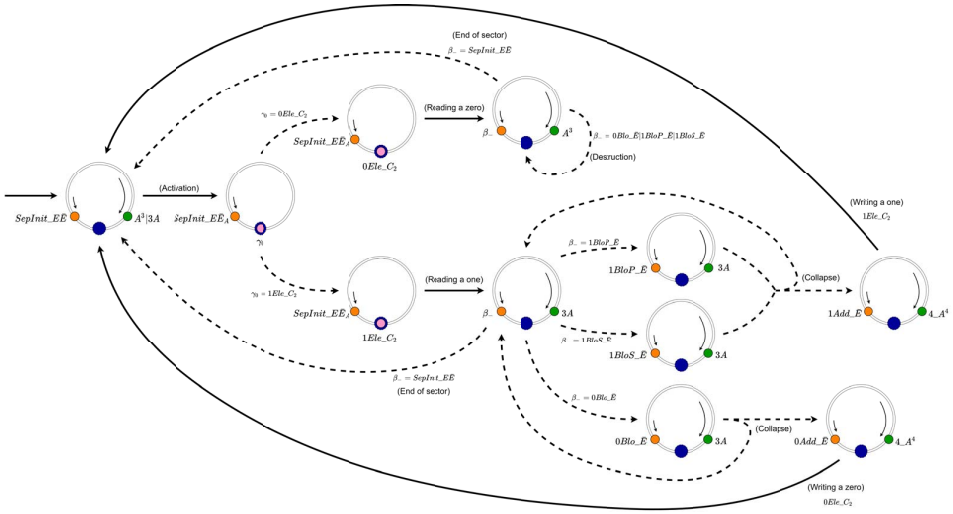


Fig. 5: Transition diagram for the operations of the CTS simulator.

the operations in the computation of the CTS. Figure 5 shows the transition diagram for the simulation, originally presented in [4].

It is important to notice that all of the particle packages that compose a section possess the same horizontal velocity. Strictly speaking, the right section has a particle A^3 that doesn't match in velocity with the rest of the section, but as it interacts with the first $SepInit\_E\bar{E}$ right away in order to activate it without changing the later's velocity, it will not be taken into account. As such, each section can move freely as long as it doesn't interact with any other section. This behavior makes it possible to isolate each section in an individual space, with all of its particles moving indefinitely in a cyclotron. The three independent cyclotrons may then be combined into one single space, a super-cyclotron. The spaces interact through contact points, carefully synchronized in order to transmit particles in resemblance of a classical virtual collider.

- The left ring injects positive $4A^4$ particle trains to the central ring.
- The right ring injects negative E and $\bar{E}$ particle sectors to the central ring.
- The central ring possesses the initial value to process, as well as the main space for all the collisions in the system after injection from both rings.

The configuration of the collision system is described by the 7-tuple:

$$(S_l, S_c, S_r, i_l, i_r, i_{cl}, i_{cr})$$

Where:

- $S_l \in \Sigma^+$ is the initial configuration for the left ring.
- $S_c \in \Sigma^+$ is the initial configuration for the central ring.
- $S_r \in \Sigma^+$ is the initial configuration for the right ring.
- $i_l \in \{0, \ldots, |S_l| - 1\}$ is the contact point from the left ring to the central ring.
- $i_r \in \{0, \ldots, |S_r| - 1\}$ is the contact point from the right ring to the central ring.
- $i_{cl}, i_{cr} \in \{0, \ldots, |S_c| - 1\}$ is the contact point from the central ring to the left and right rings, respectively.

For each contact point $i \in \{i_l, i_r, i_{cl}, i_{cr}\}$ corresponding to a ring with initial configuration S_i, the exchange of particles is carried out between cells i and $i - 1 \bmod |S_i|$. We define the neighborhoods for the contact points as:

$$\begin{aligned} N(i_r) &= (i_{cr} - 1 \bmod |S_c|, i_r, i_r + 1 \bmod |S_r|) \\ N(i_{cr}) &= (i_r - 1 \bmod |S_r|, i_{cr}, i_{cr} + 1 \bmod |S_c|) \\ N(i_l) &= (i_{cl} - 1 \bmod |S_c|, i_l, i_l + 1 \bmod |S_l|) \\ N(i_{cl}) &= (i_l - 1 \bmod |S_l|, i_{cl}, i_{cl} + 1 \bmod |S_c|) \end{aligned}$$

For convenience, we also assume that $i_{cl} \neq i_{cr}$. Figure 6 shows the operation of the virtual collider.

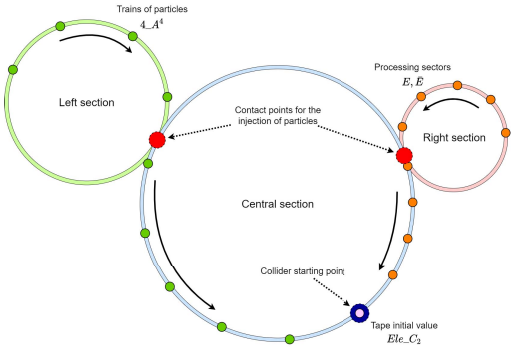
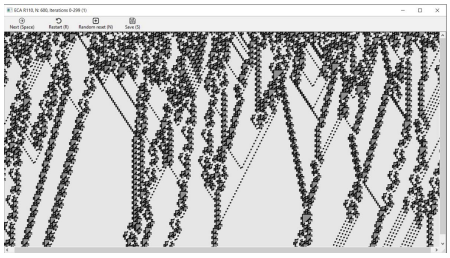


Fig. 6: Rule 110 particle collider diagram.

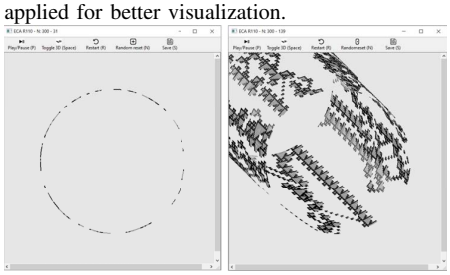
V. Program Implementation

In this research, a C++ program was implemented for the visualization of ECA cyclotrons, as well as for the execution of a virtual collider with the particles of rule 110. This project is named *CAVCollider*.

CAVCollider consists of two main modules. The first module is a CA visualizer: users select a rule for the ECA and



(a) Grid representation of the space for the selected configuration of ECA rule 110. A filter is applied for better visualization.



(b) Ring representation of the space for the selected configuration of ECA rule 110 generated in real time, in 2D and 3D mode, respectively.

Fig. 7: Screenshots of the *CAVCollider* module for elementary CA visualization.

set its initial configuration through regular expressions. The program then shows the CA in a limited grid. Alternatively, the program can process the CA in real time as a ring that simulates the behavior of a classic cyclotron. If the CA is using rule 110, the program also filters out the mosaics so as to better visualize the particles moving alongside the ring. Figures 7a and 7b show the functionality of this module.

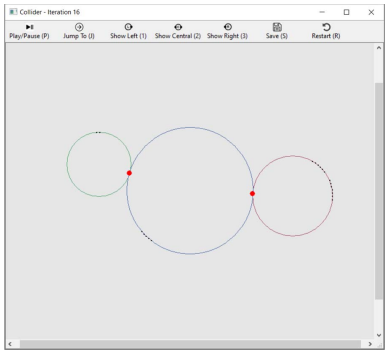


Fig. 8: Screenshot of the *CAVCollider* module for rule 110 particle collider.

The second module of *CAVCollider* is a virtual collider: users set the initial configurations for each of the 3 rings of the system, using elementary rule 110, as well as the position of the contact points for the transmission of particles. The program then executes the corresponding system, filtering the rule 110 mosaics it finds. The collider can be set to turn off the transmission of particles via contact points or the execution of any of the rings, if the user so desires. Figure 8 show the functionality of this module.

The implementation for *CAVCollider* is contained in the following GitHub project: https://github.com/olbrlvalbt/CAVCollider. The project contains a directory, *Installer*, where a ZIP file can be found. Download, extract and execute this file to install the program. Windows 10 at 64 bits and at least 8GB of memory are recommended.

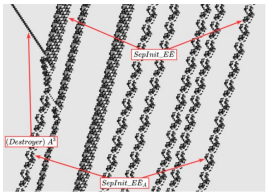
VI. Description of Collisions

To understand the operation of the CTS simulator, we will analyze the interactions between particle packages in detail.

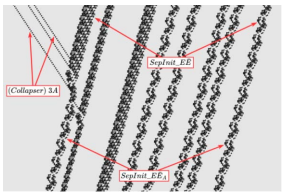
Activation of a processing sector

A sector needs to be activated in order to process values from the tape. To achieve this, a leader package $SepInit\_E\bar{E}$ must collide with one of the two activator particles A^3 and 3 A. Figures 9a and 9b show these interactions in an isolated space. The routings describing these processes are:

$$(A^3 \Leftrightarrow SepInit\_E\bar{E}) \rightarrow (SepInit\_E\bar{E}_A),$$
$$(3\ A \Leftrightarrow SepInit\_E\bar{E}) \rightarrow (SepInit\_E\bar{E}_A)$$



(a) Activation by A^3.



(b) Activation by 3 A.

Fig. 9: Activation of a processing sector.

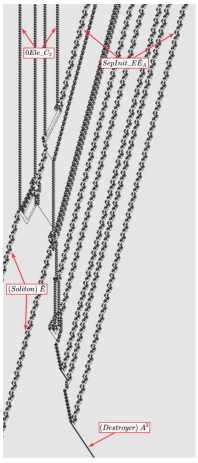
We will later see that the difference between activators originates from the generation of destroyer and collapser particles.

Reading from the tape

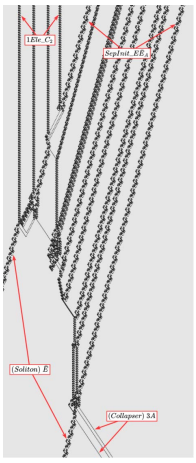
An activated leader particle $SepInit\_E\bar{E}_A$ interacts with element packages Ele_C_2, simulating the reading and deletion of a value from the tape.

Reading a zero: If the activated leader collides with a $0Ele\_C_2$, the sector enters destruction mode, preventing the creation of tape values through the generation of a destroyer particle A^3. Two soliton particles $\bar{E}$ are generated as a result of the collision. Figure 10a shows this interaction. The routing that describes this process is:

$$(0Ele\_C_2 \Leftrightarrow SepInit\_E\bar{E}_A) \rightarrow (\bar{E}, \bar{E}, A^3)$$



(a) Reading a zero from a $0Ele\_C_2$.



(b) Reading a one from a $1Ele\_C_2$.

Fig. 10: Reading of values from the tape.

Reading a one: If the activated leader collides with a $1Ele\_C_2$, the sector enters collapse mode, transforming each of its blocks $Blo\_\bar{E}$ into addition packages $Add\_\bar{E}$ through the generation of a collapser particle 3 A. Two soliton particles $\bar{E}$ are generated as a result of the collision. Figure 10b shows this interaction. The routing that describes this process is:

$$(1Ele\_C_2 \Leftrightarrow SepInit\_E\bar{E}_A) \rightarrow (\bar{E}, \bar{E}, 3\ A)$$

Destruction of blocks

When a sector enters destruction mode, all of its blocks will be eliminated. After this, the only remaining particle will be the original destroyer A^3, which will serve as activator for the next sector.

Destruction of a zero block: The routing that describes the destruction of a block $0Blo\_\bar{E}$ is:

$$(A^3 \Leftrightarrow 0Blo\_\bar{E}) \rightarrow (A^3)$$

Figure 11a shows this interaction.

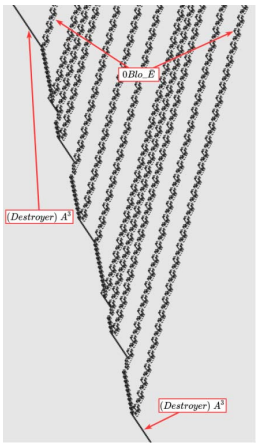
Destruction of a one block: The routing that describes the destruction of a primary block $1BloP\_\bar{E}$ is:

$$(A^3 \Leftrightarrow 1BloP\_\bar{E}) \rightarrow (A^3)$$

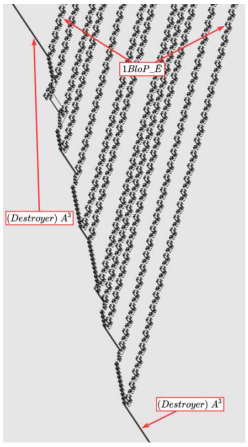
Figure 11b shows this interaction. The process is identical for the destruction of a standard block $1BloS\_\bar{E}$.

Collapsing of blocks

When a sector enters collapse mode, all of its blocks will be transformed into addition packages. After this, the only remaining particle will be the original collapser 3 A, which will serve as activator for the next sector.



(a) Destruction of a $0Blo\_\bar{E}$.



(b) Destruction of a $1BloP\_\bar{E}$.

Fig. 11: Destruction of block packages.

(a) Collapse of a $0Blo\_\bar{E}$.

(b) Collapse of a $1BloP\_\bar{E}$.

Fig. 12: Collapse of block packages.

Collapse of a zero: The routing that describes the collapse of a block $0Blo\_\bar{E}$ is:

$$(3\ A \Leftrightarrow 0Blo\_\bar{E}) \rightarrow (0Add\_\bar{E}, 3\ A)$$

Figure 12a shows this interaction.

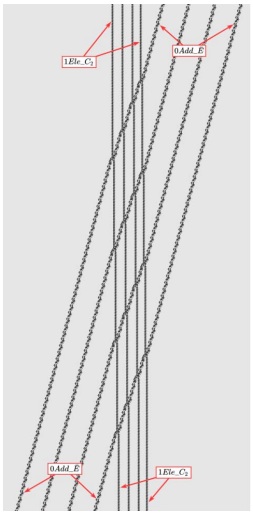
Collapse of a one: The routing that describes the collapse of a primary block $1BloP\_\bar{E}$ is:

$$(3\ A \Leftrightarrow 1BloP\_\bar{E}) \rightarrow (1Add\_\bar{E}, 3\ A)$$

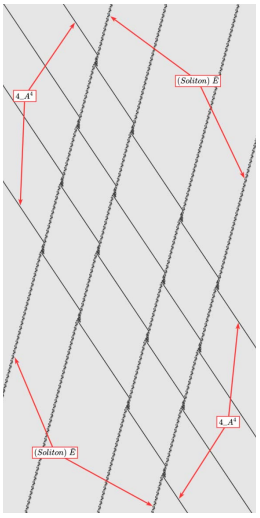
Figure 12b shows this interaction. The process is identical for the collapse of a standard block $1BloS\_\bar{E}$.

Solitonic interactions

Many particles must act as solitons in specific cases in order to keep the synchronization of the system.

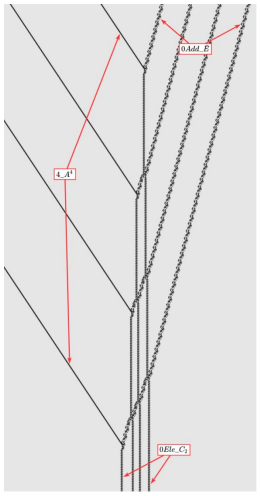


(a) Addition package $0Add\_\bar{E}$ as soliton.

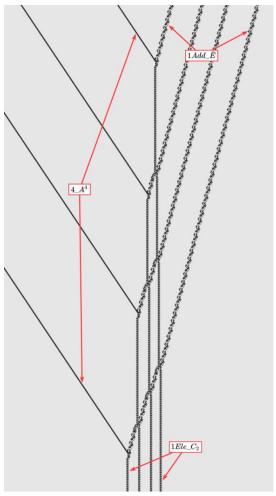


(b) Residual particles $\bar{E}$ as solitons.

Fig. 13: Solitonic interactions.



(a) Writing a zero from a $0Add\_\bar{E}$.



(b) Writing a one from a $1Add\_\bar{E}$.

Fig. 14: Writing of values to the tape.

Addition packages as solitons: Occasionally, addition packages $Add\_\bar{E}$ need to interact solitonically with element packages Ele_C_2 so as to not distort values that have been previously added to the tape. The routings are described with the routings:

$$(iEle\_C_2 \Leftrightarrow jAdd\_\bar{E}) \rightarrow (iEle\_C_2, jAdd\_\bar{E})$$

Where $i, j \in \{0, 1\}$. Figure 13a shows the solitonic interaction of a package $0Add\_\bar{E}$ with an element $1Ele\_C_2$.

Residual solitons: Residual $\bar{E}$ particles generated from the interactions of leader and block packages, must cross paths with trains 4_A^4 without distorting each other. Figure 13b shows this interaction. The reaction is described with the following routing:

$$(4\_A^4 \Leftrightarrow \bar{E}) \rightarrow (4\_A^4, \bar{E})$$

Writing to the tape

The result of collapsing blocks $Blo\_\bar{E}$ is the production of addition packages $Add\_\bar{E}$. To transform these packages into tape values (element packages Ele_C_2), a collision with a particle train 4_A^4 must happen.

Writing a zero to the tape: The routing describing the writing of a zero to the tape is:

$$(4\_A^4 \Leftrightarrow 0Add\_\bar{E}) \rightarrow (0Ele\_C_2)$$

Figure 14a shows this interaction.

Writing a one to the tape: The routing describing the writing of a one to the tape is:

$$(4\_A^4 \Leftrightarrow 1Add\_\bar{E}) \rightarrow (1Ele\_C_2)$$

Figure 14b shows this interaction.

VII. Simulation of the CTS

The configuration of the collider for the simulation of the CTS is:

$$\begin{aligned}
S_l = {} & 4_A^4(F_3) - 87e - 4_A^4(F_2) - 87e \\
& - 4_A^4(F_1) - 87e - 4_A^4(F_3) - 173e \\
& - 4_A^4(F_2) - 87e - 4_A^4(F_1) - 87e \\
& - 4_A^4(F_3) - 87e - 4_A^4(F_2) - 389e \\
& - 4_A^4(F_1) - 87e - 4_A^4(F_3) - 87e \\
& - 4_A^4(F_2) - 87e - 4_A^4(F_1) - 87e \\
& - 4_A^4(F_3) - 86e; \\
S_c = {} & 1800e - 1880e - 1Ele_C_2(A, f_1_1) - 752e; \\
S_r = {} & A^3(f_1_1) - SepInit_E\bar{E}(C, f_3_1) \\
& - 1BloP_\bar{E}(C, f_4_1) - SepInit_E\bar{E}(C, f_3_1) \\
& - 1BloP_\bar{E}(C, f_4_1) - 0Blo_\bar{E}(C, f_4_1) \\
& - 1BloS_\bar{E}(A, f_4_1) - SepInit_E\bar{E}(E, f_3_1) \\
& - 1BloP_\bar{E}(F, f_1_1) - SepInit_E\bar{E}(E, f_3_1) \\
& - 1BloP_\bar{E}(F, f_1_1) - 0Blo_\bar{E}(F, f_1_1) \\
& - 1BloS_\bar{E}(C, f_4_1) - SepInit_E\bar{E}(F, f_1_1) \\
& - 1BloP_\bar{E}(F, f_3_1) - SepInit_E\bar{E}(F, f_1_1) \\
& - 1BloP_\bar{E}(F, f_3_1) - 0Blo_\bar{E}(F, f_3_1) \\
& - 1BloS_\bar{E}(D, f_2_1) - SepInit_E\bar{E}(H, f_2_1) \\
& - 1BloP_\bar{E}(E, f_2_1) - SepInit_E\bar{E}(E, f_1_1) \\
& - 1BloP_\bar{E}(E, f_2_1) - 0Blo_\bar{E}(E, f_2_1) \\
& - 1BloS_\bar{E}(C, f_2_1) - SepInit_E\bar{E}(G, f_2_1) \\
& - 1BloP_\bar{E}(A, f_3_1); \\
i_l = 0;\ i_r = {} & 0;\ i_{cl} = 25200;\ i_{cr} = 0
\end{aligned}$$

Figure 16 shows the state of imminent collision for the execution of this system. Each ring processes a specific cellular space: the left section uses 35,912 cells; the right section,

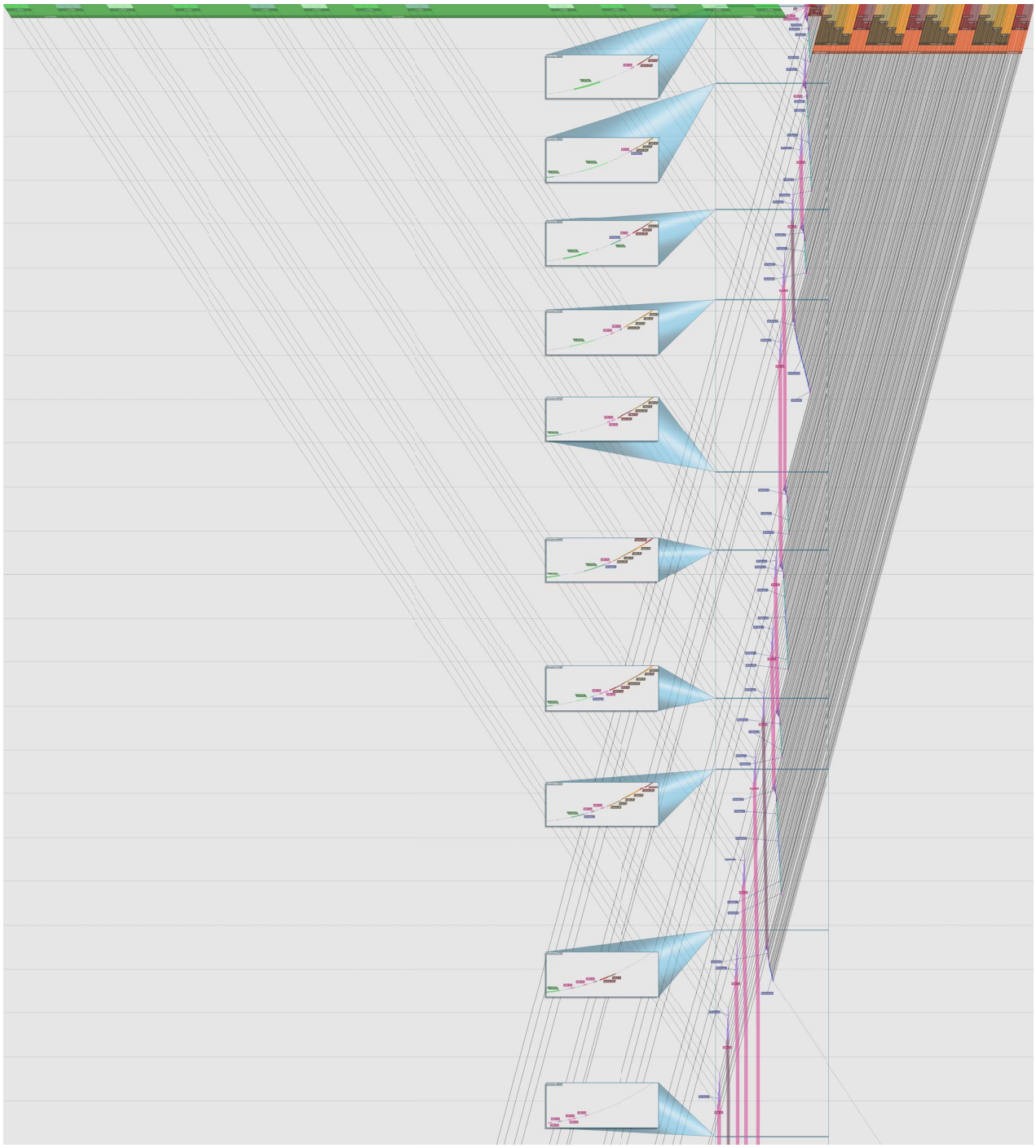


Fig. 15: Extension of the simulation of the CTS with productions $(1, 101)$ on initial tape 1 to nine transitions, the largest simulation of this system to date. The grid shows iterations 38,500 - 90,500 in a space of 46,992 cells. For each reading operation in the system, a snippet of its equivalent state in the cyclotron collider is shown.

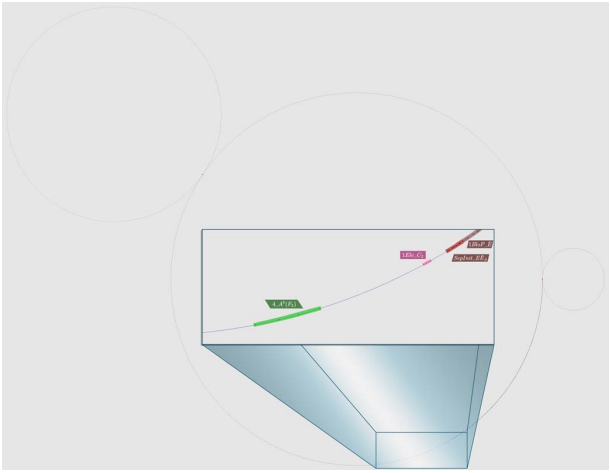


Fig. 16: State of imminent collision of the virtual collider simulating the CTS.

10,326; and the central section, 62,186. Thus, the total space for the simulator contains 108,424 cells. It is worth mentioning that the main collider consists mainly of ether in order to accomodate the particles that will be injected from the other two rings, resulting in a huge, mostly empty cellular space that otherwise would be the smallest of the three.

The simulation produces the transitions $1 \Rightarrow 1 \Rightarrow 101 \Rightarrow 011 \Rightarrow 11 \Rightarrow 11 \Rightarrow 1101 \Rightarrow 1011 \Rightarrow 011101 \Rightarrow 11101$ of the CTS. Figure 15 shows a grid that skips the process of particle injection into the main cyclotron, presenting the moment of imminent collision at iteration 38,500. The grid contains the following 52,000 iterations of the system, as well as snippets of the equivalent states in the cyclotron collider for imminent reading operations. [1]

When skipping particle injection, the simulation is able to run in a space of only 46,992 cells to execute nine transitions of the CTS, a significant improvement over the previous implementation in [4], which used 56,240 cells to execute up to five transitions of the CTS.

VIII. Conclusion

Having created a program for the codification and execution of a one-dimensional binary cellular automaton virtual collider of rule 110 particles, we were able to implement a CTS simulator by synchronizing two cyclotrons that inject particles into a main collider. This opens interesting questions and challenges, such as implementing other CTS functions and other kinds of machines. A future projection for an n-dimensional collider model would also be worth developing, as it would be able to handle a bigger universe of possibilities.

[1]For the best experience on visualizing the execution of the simulation, we recommend visiting https://drive.google.com/drive/folders/1TAsoHzuiMmVdTJFkZrcci21N9X-Ju7gA?usp=sharing, a directory containing a video that shows the collider in action with real-time labeling of the state of the particles that describe the CTS tape, as well as high quality versions of all the figures in this work. Please contact olbrlvalbt@gmail.com for inquiries on this matter.

References

[1] S. Wolfram, *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002, ISBN: 1-57955-008-8.

[2] J. Kari, *Cellular automata*, Lectures on Advanced Models of Computation, 2013.

[3] M. Cook, "Universality in elementary cellular automata," *Complex Systems: Volume 15*, 2004.

[4] G. J. Martínez, A. Adamatzky, and H. V. McIntosh, "A computation in a cellular automaton collider rule 110," *Advances in Unconventional Computing: Volume I Theory*, pp. 391–428, 2017.

[5] G. J. Martínez. "Phases fi_1 to each glider in rule 110." (2004), [Online]. Available: https://www.comunidad.escom.ipn.mx/genaro/rule110/listPhasesR110.txt. Complex Cellular Automata.

[6] G. J. Martínez. "Atlas: Collisions of gliders like phases of ether in rule 110." (2001), [Online]. Available: https://www.comunidad.escom.ipn.mx/genaro/Papers/Papers_on_CA_files/ATLAS/. Complex Cellular Automata.

[7] K. Morita, "Simple universal one-dimensional reversible cellular automata," *Journal of Cellular Automata*, 2007.

[8] K. Morita, "Simulating reversible turing machines and cyclic tag systems by one-dimensional reversible cellular automata," *Theoretical Computer Science*, 2011.

[9] E. F. Fredkin and T. Toffoli, "Design principles for achieving high-performance submicron digital technologies," *Collision-Based Computing*, pp. 27–46, 2002.

[10] N. H. Margolus, "Physics-like models of computation," *Physica D: Nonlinear Phenomena*, pp. 81–95, 1984.

[11] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora, and S. V. C. Vergara, "Reproducing the cyclic tag system developed by matthew cook with rule 110 using the phases f_1_1," *Journal of Cellular Automata*, pp. 121–161, 2010.