



# UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

## Emergent quantum phenomena from cellular automaton dynamics

*Fenomeni quantistici emergenti dalla dinamica degli automi cellulari*

Relatore

Dr. Ilaria Siloi

Correlatore

Dr. Davide Rattacaso

Laureando

Francesco Lisi

Anno Accademico 2024/2025

# Abstract

Cellular automata are simple, discrete models in which local update rules generate rich emergent behaviors. In this thesis, we investigate the quantum dynamics associated with a specific classic elementary cellular automaton by constructing a quantum many-body Hamiltonian derived from its update rules. Using a string rewriting formalism, we encode the automaton's behavior into a Hamiltonian  $H$  that governs the evolution of the quantum system. To analyze the system's behavior, we employ Tensor Network simulations, which efficiently capture the evolution of quantum states. In particular, we focus on a specific elementary cellular automaton of 25 cells, namely Rule 90, highlighting its quantum behavior in terms of localization and entanglement.

Gli automi cellulari sono semplici modelli discreti governati da regole di aggiornamento locali che generano una ricca varietà di comportamenti emergenti. In questa tesi investighiamo la dinamica quantistica relativa ad un automa elementare, costruendo un'Hamiltoniana a molti corpi derivata direttamente dalle sue regole di aggiornamento. Impiegando un formalismo di *string rewriting*, codifichiamo il comportamento dell'automa in un'hamiltoniana  $H$  che governa l'evoluzione del sistema quantistico. Infine, per analizzare il comportamento del sistema, impieghiamo simulazioni con Tensor Network, che descrivono efficacemente l'evoluzione degli stati quantistici. In particolare ci concentriamo su un automa cellulare elementare di 25 celle, ovvero il Rule 90, studiando il suo comportamento quantistico in termini di localizzazione ed entanglement.

# Contents

<b>Introduction</b>	<b>3</b>
<b>1 Cellular Automaton and String Rewriting Systems</b>	<b>4</b>
1.1 Elementary Cellular Automaton . . . . .	4
1.1.1 Definition of a Cellular Automaton . . . . .	4
1.1.2 Diagram Representation of Cellular Automata . . . . .	5
1.1.3 Wolfram’s Classification . . . . .	6
1.2 String Rewriting System . . . . .	8
1.2.1 SRS: Definition and Application to Cellular Automaton . . . . .	8
<b>2 Hamiltonian of the State Transition Diagram</b>	<b>12</b>
2.1 Continuous-Time Quantum Walk on a diagram . . . . .	12
2.1.1 From the classical representation to the computational basis . . . . .	12
2.1.2 Continuous-time quantum walk . . . . .	13
2.1.3 Computational cost of a direct construction . . . . .	14
2.2 Construction of the Hamiltonian via SRS . . . . .	14
2.2.1 Pointer Encoding and SRS . . . . .	14
2.2.2 Construction of $H$ . . . . .	16
<b>3 Simulations</b>	<b>17</b>
3.1 Simulation Method . . . . .	17
3.1.1 Matrix Product State Approximation . . . . .	17
3.1.2 Time Discretization and $\Delta t$ . . . . .	19
3.2 Observed Quantities in the Simulations . . . . .	19
3.2.1 Local Quantities . . . . .	19
3.2.2 IPR . . . . .	20
3.2.3 Maximum Bond Entropy . . . . .	21
3.2.4 Current . . . . .	21
3.3 Initial conditions . . . . .	23
3.3.1 Computational-basis state . . . . .	23
3.3.2 Wave packet . . . . .	24
3.4 Simulation results . . . . .	24
3.4.1 Computational basis state . . . . .	25
3.4.2 Wave packet . . . . .	26
3.4.3 Comparison of local quantities . . . . .	27
<b>Conclusion</b>	<b>29</b>

# Introduction

A cellular automaton is a discrete model composed of a set of cells that are updated simultaneously by applying simple local rules [1]. These rules determine the evolution of the automaton and, despite their simplicity, give rise to complex phenomena such as self-organization or chaotic behavior, modeling different processes in various fields such as physics, biology, mathematics, and computational sciences [2, 3]. In recent years, there has been a growing interest in the study of quantum versions of cellular automata, since this approach offers new perspectives on the relationship between classical and quantum computation [4, 5]. In particular, quantum cellular automata not only inherit part of the emergent complexity of classical automata, but can also display new emergent phenomena arising from quantum effects, such as correlations and superposition.

While recent literature has largely focused on quantum cellular automata constructed from scratch, in this work we map a classical elementary cellular automaton onto a many-body quantum system. We first promote the classical configurations to states of the computational basis of a quantum system. Then, building on recent developments on the quantization of string rewriting systems [6], we construct a local Hamiltonian  $H$  directly from the local rules of the automaton, embedding the irreversible dynamics of the classical automaton in a reversible quantum system governed by  $H$ . The locality of the Hamiltonian allows numerical simulations on classical computers, and opens the path for a physical implementation in quantum simulators [7].

Once the Hamiltonian is defined, the initial state of the system evolves according to the Schrödinger equation, giving rise to a *quantum walk* in the space of classical states of the automaton. The quantum walk explores the automaton configurations both forward and backward in time, that is, in a way that is intrinsically different from the evolution of a classical machine. This approach offers new perspectives on the relationship between classical and quantum computation: for example, such a system could be used to reconstruct the original states of a machine, or to study the time required for a quantum system to reach a halting state that terminates the computation.

We numerically simulate the dynamics generated by a quantized version of a specific elementary cellular automaton, named Rule 90. Two different initial conditions are considered in the simulations of the system evolution: a single state of the computational basis and a Gaussian wave packet constructed as a superposition of multiple basis states. For both conditions, we observe and compare the spatial distribution of local observables and global quantities, such as the *Inverse Participation Ratio* (IPR), the *Maximum Bond Entropy*, and the current. This analysis is carried out using simulation methods based on *Tensor Networks*, in particular the Matrix Product State (MPS) representation, which allows us to efficiently encode the states of a quantum many-body system.

The simulation results show that the choice of the initial condition is decisive in the evolution of the system, leading to very different behaviors in terms of localization and quantum correlations. In particular, when the quantum system is prepared in a time-reversal-breaking state, i.e. with the wave packet propagating in one direction, it can effectively simulate the irreversible evolution of the original automaton.

# Chapter 1

## Cellular Automaton and String Rewriting Systems

*In this chapter, we introduce the concept of cellular automata and the String Rewriting System (SRS) formalism. We first define cellular automata as discrete models with local evolution rules that, despite their simplicity, give rise to complex emergent phenomena. We then show how to represent the dynamics of automata through state diagrams and how to classify them based on their emergent behaviors. Finally, we introduce the SRS formalism, which allows us to simulate the global evolution of a cellular automaton by using local transformations.*

### 1.1 Elementary Cellular Automaton

Cellular automata are discrete models composed of a set of cells that are updated simultaneously applying simple local rules. Despite their apparent simplicity, these systems exhibit complex emergent phenomena, such as the formation of ordered structures (self-organization), the propagation of information, and the generation of chaotic behaviors [1]. These characteristics make cellular automata useful in numerous scientific fields: from the modeling of physical systems to biology [2, 3] (e.g., tissue growth, epidemic spreading), from the simulation of complex and self-organizing systems to computational theory, where some cellular automata have been proven to be Turing-complete. In essence, their ability to generate global behaviors from simple local interactions makes them fundamental tools for the study of complexity [6].

#### 1.1.1 Definition of a Cellular Automaton

A *cellular automaton* is a discrete dynamical system defined on a regular  $d$ -dimensional lattice of  $N \in \mathbb{N}$  cells, in which each cell  $s_i$  ( $i = 1, \dots, N$ ) takes one of  $k$  discrete values [1]:

$$s_i(t) \in \{\sigma_j\}_{j=1,\dots,k}, \quad \sigma_j \in \mathbb{Z}.$$

Its dynamics is governed by a local deterministic rule  $\Phi$ , which is a discrete function that updates the value of each cell at every time-step:

$$s_i(t+1) = \Phi(s_{i-r}(t), \dots, s_i(t), \dots, s_{i+r}(t)),$$

where the value of  $s_i(t+1)$  also depends on the states of its neighbors within a radius  $r$ . Hence, given the automaton configuration at time  $t$ , its configuration at time  $t+1$  is obtained by applying the rule to all cells at the same time. A particularly important class is that of the *elementary cellular automaton*, defined by choosing a dimension  $d = 1$ . In this work we focus on this class assuming  $k = 2$  (binary states) and a radius  $r = 1$ , which means that each cell can take only two possible states ( $\pm 1$ ) and its next state depends on the cell itself and its two nearest neighbors:

$$s_i(t) \in \{-1, 1\}, \quad s_i(t+1) = \Phi(s_{i-1}(t), s_i(t), s_{i+1}(t)).$$

Therefore, since each cell has two states, the number of possible inputs for a rule  $\Phi$  is  $2^3$ , while the output is binary: in other words, there are  $2^{2^3} = 256$  distinct possible definitions of  $\Phi$ , commonly referred to as Rule 0 through Rule 255 in Wolfram's notation [1]. For the boundary cells, namely, the first and the last one, we assume the presence of a virtual neighbor in state  $-1$  on the left and on the right, respectively. In this work, we focus on this class of cellular automata, paying particular attention to Rule 90.

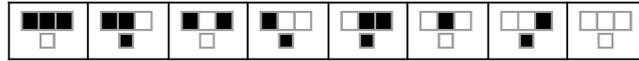


Figure 1.1: Rule 90: on top the input cells for the rule, on bottom the output after 1 time-step. We represent the state  $+1$  in black and the state  $-1$  in white. [8]

As we will see in the following sections, it is interesting to notice that, despite the simplicity of this construction, the automaton exhibits rich and different dynamical behaviors, and, looking at the automaton evolution, a remarkable variety of highly complex patterns emerges.

### 1.1.2 Diagram Representation of Cellular Automata

A *directed graph* is a mathematical structure defined as  $G = (V, E)$ , where  $V$  is a set of *nodes* (or *vertices*) and  $E \subseteq V \times V$  is a set of *arcs* (or *links*) connecting them, which may have a direction. Graphs are widely used to represent networks, relationships between objects or states, and discrete dynamical systems, as in our case.

In fact, each configuration of an elementary cellular automaton can be represented as a string of  $N$  binary characters  $s_i$ , and a state of the  $2^N$  configurations can be assigned to a node  $V$ . Then, two configurations that differ by a single application of the rule are vertices connected by an arc. The graph constructed in this way from a cellular automaton is called *state-transition diagram*. Since the update function  $\Phi$  is deterministic, every node has only one outgoing arc, but multiple nodes can point to the same subsequent state, showing also the irreversibility of the process.

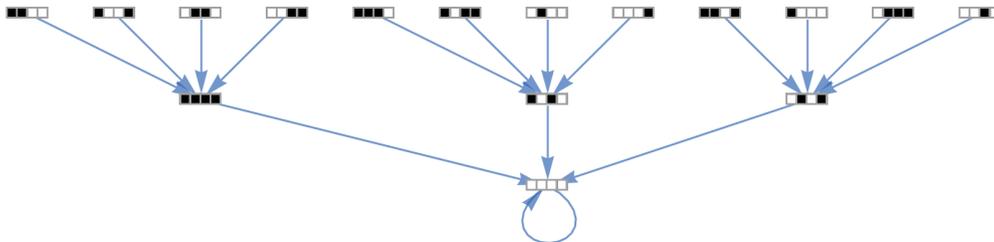


Figure 1.2: Example of a state-transition diagram for Rule 90 with  $N = 4$  cells. Each node represents one of the  $2^4 = 16$  states, and arcs show how states evolve under the rule [8].

We also observe that there are some configurations to which the system can converge after some finite number of time-steps and we call them *attractors*. For cellular automata, attractors appear in two forms:

- **Fixed points:** a single configuration that maps to itself.
- **Cycles:** a finite sequence of configurations that repeat periodically.

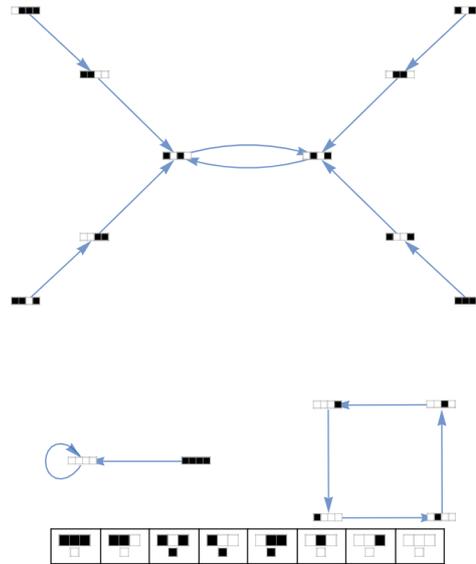
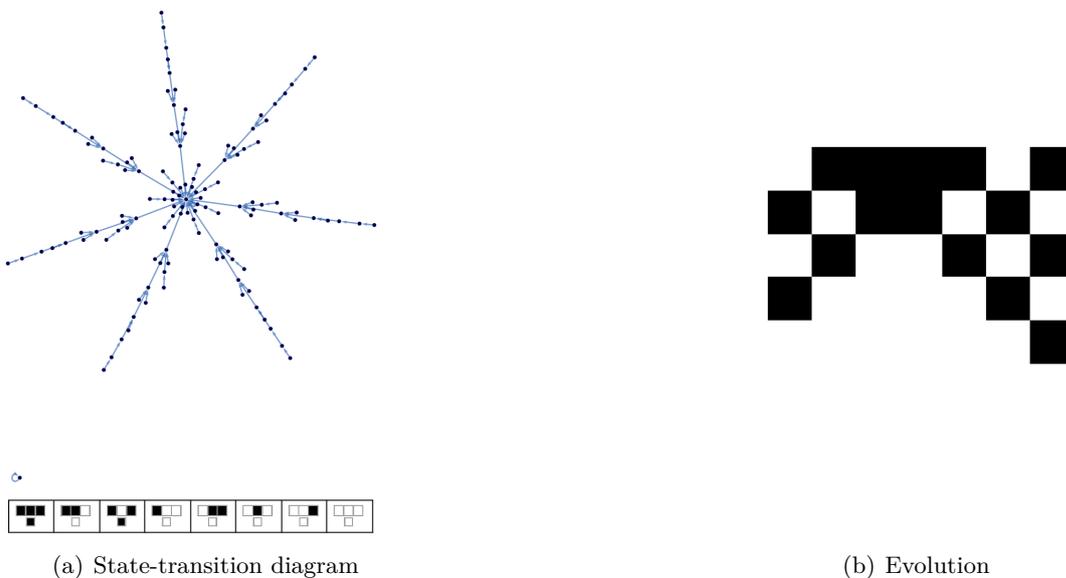


Figure 1.3: State transition diagram of Rule 56 with 4 cells: the disconnected graph is an example of a fixed point and two cycles composed by two and four configurations [8].

### 1.1.3 Wolfram’s Classification

In 1984, Stephen Wolfram proposed an empirical classification of cellular automata based on their long-term dynamical behaviors [1, 9]. In particular, for an elementary cellular automata each rule belongs to one of four qualitative classes. As we illustrate here the distinct dynamical behaviors of automata in each class are also reflected in the shapes of their state diagrams:

- **Class 1:** the automaton evolves to a homogeneous state. In other words, from almost any initial configuration, all cells converge to the same value (all 0s or all 1s). Hence, no structures persist over time, and the state-transition diagram collapses quickly to a single fixed-point attractor, in which all nodes converge.



(a) State-transition diagram

(b) Evolution

Figure 1.4: Example evolution of a Class 1 cellular automaton (Rule 160) for a lattice of size  $N = 7$ . In (b) time flows from top (initial random state) to bottom (final homogeneous state with all ‘white’ cells) [8].

- **Class 2:** these rules produce periodic or stable patterns. Namely, after a short transient, the automaton settles into simple repeating motifs, so in their diagrams there are several simple attractors.

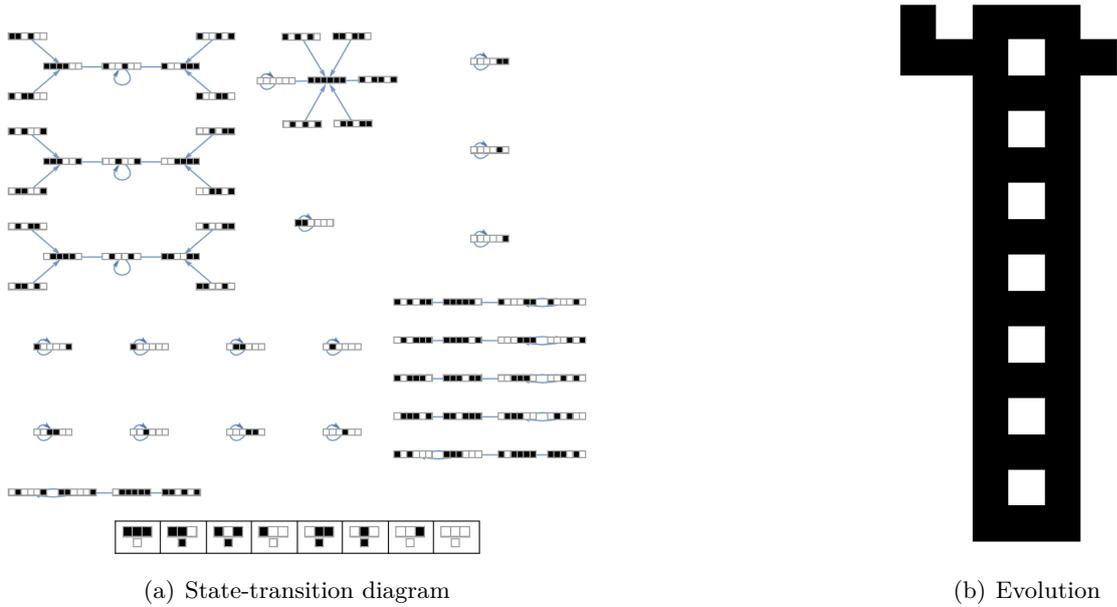


Figure 1.5: Example evolution of a Class 2 cellular automaton (Rule 108) for a lattice of size  $N = 6$ . In (b) time flows from top (initial random state) to bottom (final repeating motif) [8].

- Class 3:** The automaton exhibits chaotic behavior, even when it starts from simple initial conditions such as a single central cell in state 1 and all other cells in state  $-1$ . Its evolution often shows fractal-like structures and can be remarkably different even for small changes in the initial conditions. Thus, even if the evolution is deterministic, the behavior is chaotic, however the statistical properties of this class converge to stable averages over time. Finally, in terms of the state-transition diagram, this class is characterized by very large attractors, so that periodicity becomes evident only after long transients, reflecting its pseudorandom nature. This class includes Rule 90.

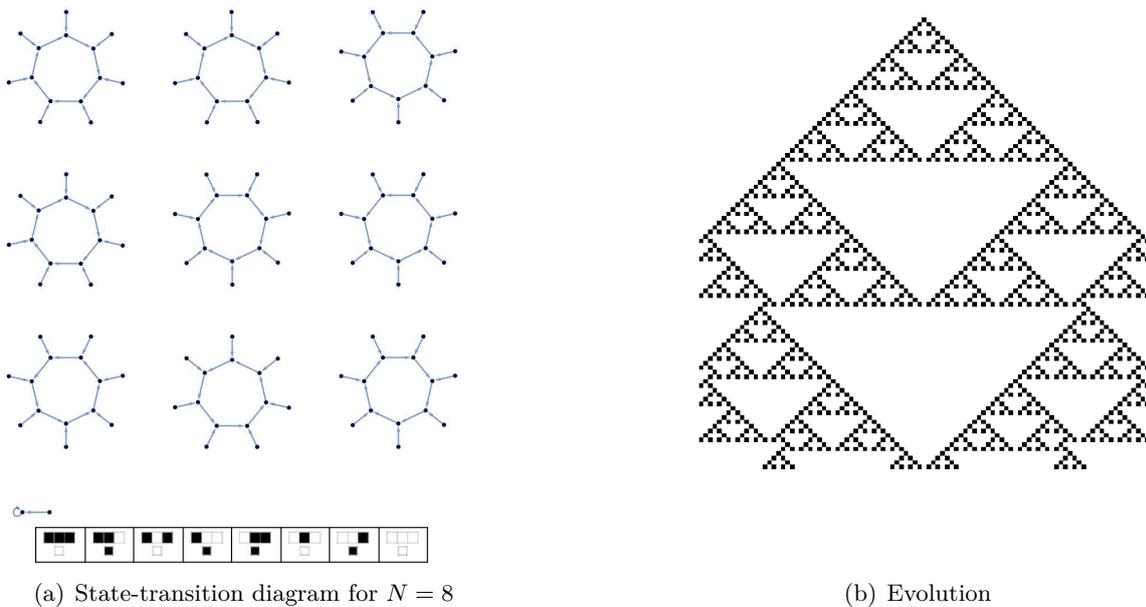


Figure 1.6: Example evolution of a Class 3 cellular automaton (Rule 90) [8]. The second image shows the evolution from an initial state of  $N = 100$  cells with a single central cell in state 1: its evolution shows a pattern known as Sierpinski triangles [9].



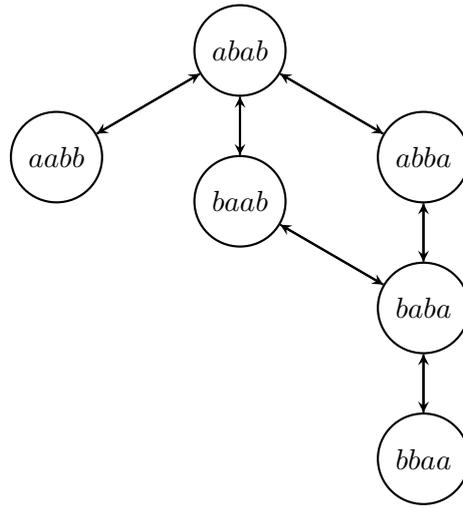


Figure 1.8: Graph of the rule  $ab \approx ba$ . From  $abab$  we can generate different sequences applying the rule in different positions.

We now introduce a string rewriting system that can simulate a cellular automaton. In other words, by applying a sequence of rewriting rules, we can transform any given state of the automaton into its next state. This transformation may involve passing through intermediate strings that are not themselves valid states of the cellular automaton, as discussed in the next paragraph.

As discussed previously, the configuration of an elementary cellular automaton can be represented as a string of states drawn from a binary alphabet. We define this binary string as *tape*:

$$s_1, s_2, s_3, s_4, \dots, s_N, \quad s_i \in \{-1, 1\}.$$

To encode the application order of the rules, we now introduce a *pointer*  $p$  that “moves” along the tape and explicitly selects the cell where the local rule has to be applied.

Formally, each cell is no longer described by a single symbol  $s_i$  of the tape but by a pair of indices  $(p_i, s_i)$

$$(p_1, s_1), (p_2, s_2), \dots, (p_N, s_N), \quad s_i \in \{-1, 1\}, \quad p_i \in \{0, \pm 1\},$$

where:

- $p_i = 0$  if the cell is not visited by the pointer and the local rule will be not applied on  $s_i$ ;
- $p_i = \pm 1$  if the pointer is present on that cell, storing the original value  $s_{i-1}$  of the cell  $i - 1$  before the update.

Hence, after applying the local rule on the visited cell  $i$ , the pointer is removed from that cell ( $p_i = 0$ ) and moved to the next one  $i + 1$ , assuming the original value of the just updated cell ( $p_{i+1} = s_i$ ).

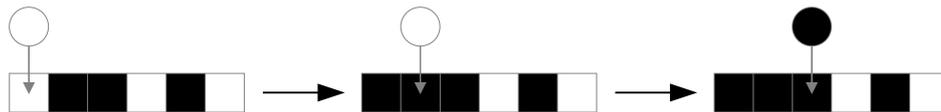


Figure 1.9: Graphical example of the pointer’s mechanism. The pointer indicates the cell the Rule 90 is applied to, and stores the “color”, i.e. the state, of the previous cell before it was updated. Once the current cell has been updated, the pointer shifts to the next cell, repeating the process.

In particular, the original automaton states, defined by the application of the local rule to all cells, are represented by configurations of the form

$$(p_1, s_1), (0, s_2), (0, s_3), \dots, (0, s_N), \quad p_1 \neq 0.$$

As a result, this representation naturally leads to an extended alphabet  $A$ : each cell is no longer described by a single binary variable  $\{\pm 1\}$ , but includes all combinations of the binary variable  $\{\pm 1\}$  and the state of the pointer  $\{0, \pm 1\}$  (two-index representation). Then, the pristine binary alphabet can be mapped to set of integers:

$$\{0, 1, 2, 3, 4, 5\} \equiv \{(0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)\},$$

On the other hand, the rewriting rules now incorporate both the automaton local update and the pointer shift, assuming the following form:

$$R \ni r_i : ((p_i, s_i), (0, s_{i+1})) \longrightarrow ((0, \Phi(p_i, s_i, s_{i+1})), (s_i, s_{i+1})),$$

where the initial pointer value  $p_i$  equals the original state of the previous cell  $s_{i-1}$  and the function  $\Phi$  is exactly the original local rule, taking as input  $(s_{i-1}, s_i, s_{i+1})$  and returning the updated value at the cell  $i$ . In this way, the update order is encoded directly into the rewriting rules themselves, and we now have a SRS that simulates the automaton. In Fig. 1.11, we show an example of the application of the SRS.

Finally, we also observe that:

- because the rules move the pointer but never create or destroy it, the number of pointers is a conserved quantity;
- from a diagram perspective, this construction replaces each arc that connects two configurations with a sequence of  $N$  smaller arcs (Fig. 1.10), each corresponding to an elementary rewriting step.

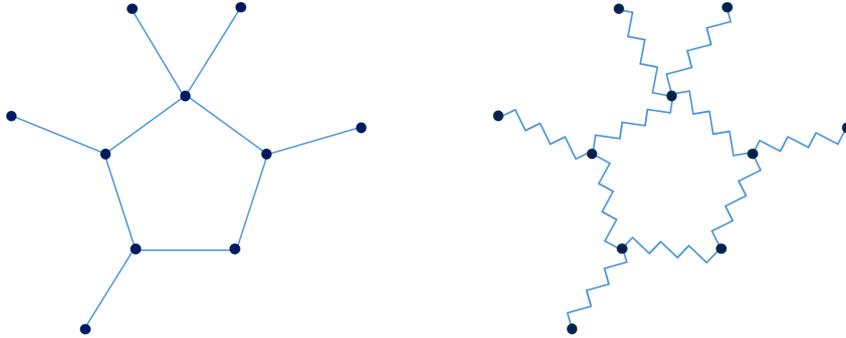


Figure 1.10: Difference between state-transition diagram and SRS: each arc is replaced by a sequence a smaller arcs

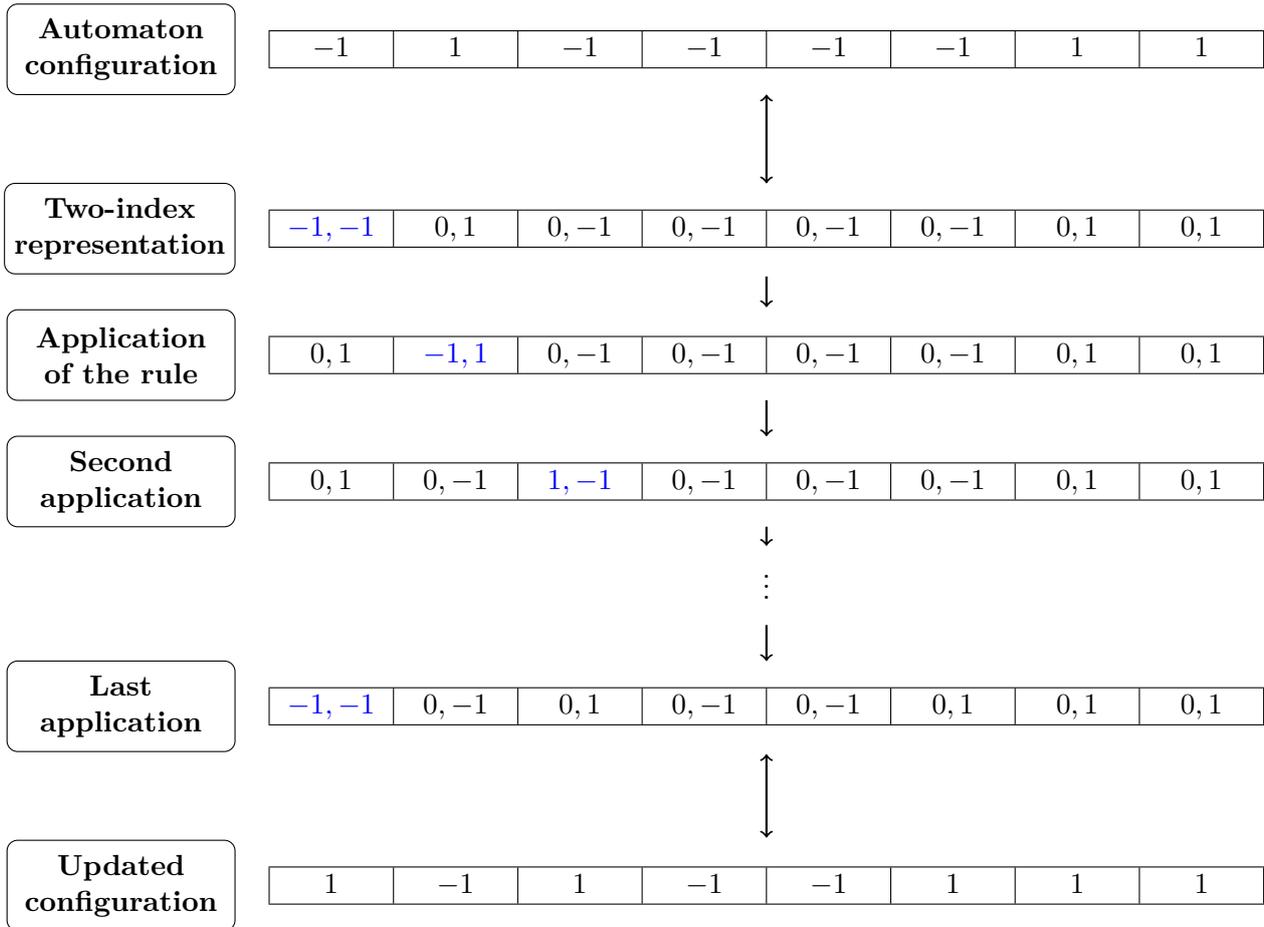


Figure 1.11: Starting from the cellular automaton string, we move to the pointer representation, apply Rule 90 iteratively (with the pointer moving through the cells), and finally recover the original updated configuration, showing the equivalence between the two descriptions. From a diagram perspective, we divide the arc between two configurations in  $N$  intermediate steps.

## Chapter 2

# Hamiltonian of the State Transition Diagram

*We want to study how the same configuration structure (i.e., the same diagram) behaves if the dynamics is quantum rather than classical. In particular, we analyze how classical configurations can be represented as states of the computational basis of a quantum register, in order to exploit quantum superposition to explore the diagram in a different way. We then introduce the Hamiltonian associated with this dynamics, showing how it reproduces the connectivity of the diagram and enables evolution through a continuous-time quantum walk. We also discuss the computational cost of a direct construction of the Hamiltonian, motivating the adoption of an alternative strategy based on local rewriting rules (SRS) and on the use of a pointer to simulate the update order of the automaton. This approach leads to a local Hamiltonian capable of reproducing the classical automaton dynamics in the quantum regime while keeping the computational complexity manageable.*

### 2.1 Continuous-Time Quantum Walk on a diagram

To model the quantum dynamics of the automaton, we represent the nodes of the diagram, i.e. strings of length  $N$ , as computational basis states of an  $N$ -qubit register and we define a Hamiltonian which maps a computational basis state into another one if and only if the two states represent two connected nodes. The evolution generated by this Hamiltonian is a continuous-time quantum walk on the state diagram and the evolution of a computational basis state results in a superposition involving only those states directly connected to it in the diagram.

Although the arcs and nodes of the diagram remain unchanged, the quantum walk differs substantially from any classical graph exploration method. In fact, a quantum walk is the quantum analogue of the classical random walk, where the dynamics of the particle (the ‘walker’) is governed by the superposition and interference between amplitudes, leading to different spreading behavior (typically ballistic instead of diffusive). This exploration, implementable on a quantum computer, provides the following advantages:

- exploration of multiple paths on the diagram;
- possibility of using a superposition of nodes as an initial state, such as a wave packet, and not only a single node;
- possibility of traversing arcs in both directions.

#### 2.1.1 From the classical representation to the computational basis

As previously defined, a configuration of the classical automaton is a binary string of length  $N$ :

$$s = (s_1, s_2, \dots, s_N), \quad s_i = \pm 1$$

We encode these strings into a system of qubits, i.e., a many-body quantum system composed of several 2-dimensional local systems. We call these local quantum systems qubits, and denote their basis vectors as  $|-1\rangle$  and  $|1\rangle$ . In quantum computing, each binary character is typically mapped to a qubit, so that the value taken by each character is mapped directly onto one of the two basis states:

$$s_i = -1 \iff |-1\rangle, \quad s_i = 1 \iff |1\rangle,$$

In this way, it is possible to represent the entire classical string as a tensor product of  $N$  qubits:

$$|s\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_N\rangle \in \mathcal{H} = (\mathbb{C}^2)^{\otimes N},$$

where the chosen basis for the Hilbert spaces  $\mathcal{H}$  is precisely the set of configurations  $\{|s_1 s_2 s_3 \dots s_N\rangle\}$ . In other words, each classical string is in one-to-one correspondence with a basis state of our  $N$ -qubit quantum register, avoiding loss of information in the transition from classical to quantum. The advantage is that once this correspondence is established it is possible to exploit quantum superposition to perform operations on strings that are not possible classically.

### 2.1.2 Continuous-time quantum walk

Once the nodes of the diagram have been represented as orthonormal states on the computational basis, we want to explore it starting from one node and reaching nodes in more distant regions.

In the classical cellular automaton, the dynamics is fully deterministic: from each node there is only one outgoing arc, and the exploration proceeds along a unique, well-defined path leading from one configuration to its single successor.

In contrast, the path on the diagram can now be defined by a *continuous-time quantum walk* (CTQW) [10], namely a dynamics governed by a Hamiltonian  $H$  that encodes the structure of the diagram itself. In this way, the evolution is no longer restricted to a single deterministic path, because quantum superposition allows the simultaneous exploration of multiple compatible paths.

More specifically, to implement a continuous-time quantum walk we can construct a Hamiltonian of the form

$$H = \sum_{i,j} A_{ij} (|s^{(i)}\rangle\langle s^{(j)}| + |s^{(j)}\rangle\langle s^{(i)}|), \quad (2.1)$$

where  $|s^{(i)}\rangle$  is a node of the diagram, i.e. a basis state  $|s^{(i)}\rangle = |s_1^{(i)} s_2^{(i)} \dots s_N^{(i)}\rangle$ , while  $A$  is the adjacency matrix of the diagram, defined as

$$A_{ij} = \begin{cases} 1 & \text{if there exists an arc between the nodes } |s^{(i)}\rangle, |s^{(j)}\rangle \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

In essence, this Hamiltonian consists only of hopping terms between connected diagram states: when applied to a basis state, it yields a superposition involving only the states adjacent to it, namely

$$H|s^{(i)}\rangle = |s^{(j)}\rangle + |s^{(k)}\rangle + \dots,$$

where  $|s^{(j)}\rangle, |s^{(k)}\rangle, \dots$  are the nodes connected to  $|s^{(i)}\rangle$  by an arc of the diagram.

Once  $H$  is defined, the evolution of a generic quantum state  $|\psi(t)\rangle \in \mathcal{H}$  is governed by the Schrödinger equation:

$$i \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle, \quad |\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle.$$

Thus, even when starting from a single node,  $|\psi(0)\rangle = |s^{(i)}\rangle$ , the evolved state  $|\psi(t)\rangle$  becomes a superposition of nodes of the diagram, with complex probability amplitudes. This behavior can give rise to phenomena absent in the classical case, such as interference effects, making the resulting dynamics particularly interesting to study.

Finally, an important aspect of the continuous-time quantum walk should be noted: since the Hamiltonian is Hermitian, the diagram is effectively *undirected*. Indeed, if there is a hopping term  $|s^{(i)}\rangle\langle s^{(j)}|$  from node  $s^{(i)}$  to node  $s^{(j)}$ , there is also the corresponding operator allowing the reverse transition,  $|s^{(j)}\rangle\langle s^{(i)}|$  or equivalently, because the evolution is unitary, the process is reversible. This means that motion along the diagram is possible both “forward”, as in the classical case, and “backward”.

### 2.1.3 Computational cost of a direct construction

The main limitation of this approach is that the direct implementation of the Hamiltonian in Eq. (2.1) requires the construction of a matrix of size  $2^N \times 2^N$ , with a computational cost that grows exponentially with the system size  $N$ . This reflects the fact that the transition between two configurations in a cellular automaton is a global update and, therefore, this representation fails to provide a description in terms of local Hamiltonian, i.e., as a sum of Hermitian operators acting in parallel on finite regions of the configuration.

Moreover, performing this construction requires prior knowledge of the connections between all nodes of the diagram, which may be highly branched and complex. For these reasons, this construction becomes impractical even for moderate values of  $N$ , motivating the search for an alternative representation of  $H$  based on *local rewriting rules*. Finally, we also observe that a local Hamiltonian will be of particular interest since it is closer to the physical systems observed in nature.

## 2.2 Construction of the Hamiltonian via SRS

As discussed above, it is not convenient to write the Hamiltonian  $H$  in terms of operators  $|s^{(i)}\rangle\langle s^{(j)}|$  that simultaneously update all cells defining the state of the automaton.

Instead, it is more convenient to update one cell at a time, writing the Hamiltonian in terms of local operators, i.e., operators acting on at most  $w$  adjacent qubits, where  $w$  does not depend on the system size. These operators encode the action of rewriting rules on the computational basis states. In particular, the action of the rule  $r$ , as defined in Chapter 1, is:

$$r = (\alpha_i \dots \gamma_{i+w-1} \dots \approx \alpha'_i \dots \gamma'_{i+w-1}),$$

and is encoded by the local operator

$$\hat{r} = \mathbb{I} \otimes \dots \otimes \mathbb{I} \otimes |\alpha'_i \dots \gamma'_{i+w-1}\rangle\langle \alpha_i \dots \gamma_{i+w-1}| \otimes \mathbb{I} \otimes \dots \otimes \mathbb{I},$$

which acts non-trivially only on the qubits at position from  $i$  to  $i + w - 1$ , updating them accordingly to the automaton rule.

When constructing a local Hamiltonian to simulate the automaton dynamics, however, we have also to encode the fact that the update rule is applied in a specific order, from the leftmost cell to the rightmost one. Simply summing over the local operators  $\hat{r}_i$  would update all matching positions simultaneously unless one modifies the operators “from the outside” at each local application, effectively rendering the Hamiltonian time-dependent.

To overcome this limitation, we introduce a pointer and employ the SRS formalism to encode in  $H$  the order of application of the local rules, keeping track of the cells already updated so that the rule can be reapplied consistently.

### 2.2.1 Pointer Encoding and SRS

As explained in Chapter 1, we associate with each cell a pair of indices encoding both the cell state (stored in the tape) and the presence of a pointer (indicating where to apply the rule and storing information about the previous cell state):

$$(p_i, s_i), \quad p_i = 0, \pm 1, \quad s_i = \pm 1,$$

which we map, for simplicity, to the compact alphabet

$$\{0, 1, 2, 3, 4, 5\} \equiv \{(0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)\}.$$

We therefore again deal with strings of length  $N$ , but now with non-binary characters. Nevertheless, we can proceed as before by representing these strings as states of the computational basis, with the difference that each cell is now associated with a *qudit*, that is, in our case, a quantum system with six basis states:

$$(p_i = 0, s_i = -1) \Leftrightarrow |0\rangle \quad (p_i = 0, s_i = 1) \Leftrightarrow |1\rangle, \quad \dots, \quad (p_i = 1, s_i = 1) \Leftrightarrow |5\rangle.$$

Each string is thus in one-to-one correspondence with a basis state of the form

$$|p, s\rangle = |p_1, s_1\rangle \otimes |p_2, s_2\rangle \otimes \dots \otimes |p_N, s_N\rangle \in \mathcal{H} = (\mathbb{C}^6)^{\otimes N}.$$

The next step is to represent the SRS rules as operators  $\hat{r}$ ; that is, we wish to reproduce on the subsystem  $|p_i s_i\rangle \otimes |p_{i+1} s_{i+1}\rangle$  the same transformation that the rules produce on the substrings  $((p_i, s_i), (p_{i+1}, s_{i+1}))$ . In particular, since the rules of the SRS act as

$$r_i : ((p_i, s_i), (0, s_{i+1})) \longrightarrow ((0, \Phi(p_i, s_i, s_{i+1})), (s_i, s_{i+1})), \quad p_i = \pm 1, \quad (2.3)$$

we define a single local operator  $\hat{r}_i$  incorporating all possible transitions:

$$\hat{r}_i = \sum_{p_i, s_i, s_{i+1}} \mathbb{I} \otimes \dots \otimes |0, \Phi\rangle \langle p_i, s_i| \otimes |s_i, s_{i+1}\rangle \langle 0, s_{i+1}| \otimes \dots \otimes \mathbb{I}, \quad (2.4)$$

where the operator acts non-trivially only on cells  $i$  and  $i + 1$ , while acting as the identity on all other cells.

**Example.** *Rule 90 is defined by the local function*

$$\Phi(s_{i-1}, s_i, s_{i+1}) = -s_{i-1}(t) s_{i+1}(t).$$

*For example, two classical transitions are:*

$$(1, 1, 1) \longrightarrow -1, \quad (-1, 1, 1) \longrightarrow 1.$$

*Using the extended encoding with pointer, these local rules become:*

$$((1, 1), (0, 1)) \longrightarrow ((0, -1), (1, 1)), \quad ((-1, 1), (0, 1)) \longrightarrow ((0, 1), (1, 1)).$$

*In numerical encoding:*

$$(5, 1) \longrightarrow (0, 5), \quad (3, 1) \longrightarrow (1, 5).$$

*These rules are implemented on cell  $i$  in the Hamiltonian as:*

$$\begin{aligned} \hat{r}_i^1 &= \mathbb{I} \otimes \dots |0, 5\rangle \langle 5, 1| \dots \otimes \mathbb{I}, & \hat{r}_i^{1\dagger} &= \mathbb{I} \otimes \dots |5, 1\rangle \langle 0, 5| \dots \otimes \mathbb{I}, \\ \hat{r}_i^2 &= \mathbb{I} \otimes \dots |1, 5\rangle \langle 3, 1| \dots \otimes \mathbb{I}, & \hat{r}_i^{2\dagger} &= \mathbb{I} \otimes \dots |3, 1\rangle \langle 1, 5| \dots \otimes \mathbb{I}. \end{aligned}$$

*And summed with the others  $\hat{r}_i^k + \hat{r}_i^{k\dagger}$  give back  $\hat{r}_i + \hat{r}_i^\dagger$ .*

### 2.2.2 Construction of $H$

It is now possible to write the Hamiltonian as a sum of the rules, since these also encode the information about their order of application:

$$H = \sum_{i=1}^{N-1} \left( \hat{r}_i + \hat{r}_i^\dagger \right),$$

where the Hermitian conjugate terms are included to ensure the Hermiticity of  $H$ . The difference with the previous formulation is that now each rule is applied only in the presence of the pointer, which is moved at each application so that, after  $N$  steps, the system evolves from one node of the diagram to another. It is therefore possible to evolve a generic initial state by applying  $e^{-iHt}$  to it.

**Example.** Consider a string of length  $N = 4$ :

$$|p, s\rangle = |(0, 1), (1, 1), (0, 1), (0, -1)\rangle \equiv |1, 5, 1, 0\rangle.$$

According to Rule 90 defined above:

$$(5, 1) \longrightarrow (0, 5), \quad (3, 1) \longrightarrow (1, 5), \quad \dots$$

With  $H$  defined as above, the only operators acting non-trivially on the state  $|p, s\rangle$  are  $\hat{r}_1^\dagger, \hat{r}_2$ :

$$H|1, 5, 1, 0\rangle = (\hat{r}_1^\dagger + \hat{r}_2)|1, 5, 1, 0\rangle = |3, 1, 1, 0\rangle + |1, 0, 5, 0\rangle.$$

The Hermiticity of  $H$  allows the pointer to move in both directions, enabling forward and backward transitions.

In summary, to make the problem computationally tractable, we introduced an additional degree of freedom—the pointer—to encode directly in a local Hamiltonian the order of the rules. From the diagrams point of view, this corresponds to dividing each arc into  $N$  segments, i.e., connecting two nodes with  $N$  intermediate nodes representing configurations in which the pointer has not yet completed scanning the tape that contains the information on the state of each cell. The fundamental difference compared to previous approaches is that the quantum walk is now performed on an *extended diagram* that includes both the original nodes and these intermediate ones (see Fig. 1.10).

Finally, we note that:

- The configuration space now has size  $4N \cdot 2^{N-1}$ : each character can take six values, but the string must satisfy the constraint of pointer conservation, i.e., the presence of exactly one pointer. Consequently, only one character can take values in  $\{2, 3, 4, 5\}$ , while the remaining ones must be in  $\{0, 1\}$ .
- This representation is computationally convenient: although the configuration space has been enlarged, the number of operators required to build  $H$  scales linearly with the system size  $N$ , as only one operator  $\hat{r}_i$  per cell needs to be defined.
- The Hamiltonian  $H$  is straightforward to implement on a quantum computer or on a quantum simulator, since it is sum of local operators.

# Chapter 3

## Simulations

In the previous chapter, we introduced a local Hamiltonian description of the automaton evolution using a pointer and a corresponding string rewriting system. Here, we perform numerical simulation of this many-body system using tensor network methods for the simulation of the quantum many-body system. First, we define the main simulation parameters: the bond dimension  $\chi$ , required to reduce the computational cost by using Tensor Network, and the time step  $\Delta t$ , necessary to solve the Schrödinger equation. Next, we introduce some quantities measured during the simulation, including local ones, computed cell by cell, and global ones, such as the IPR, which measures the localization of the system's state, and the Maximum Bond Entropy, which quantifies the entanglement generated during the evolution. Finally, we present and discuss the simulation results.

### 3.1 Simulation Method

The simulations are performed using Quantum Tea [11], a software based on tensor networks, which allows to represent quantum states as Matrix Product States (MPS) and to simulate the system's evolution through the Time-Dependent Variational Principle (TDVP) algorithm. In this section, we briefly present these methods, focusing on the main parameters that govern the simulations.

#### 3.1.1 Matrix Product State Approximation

Because our system is composed of  $N$  qudits with  $d = 6$  local states, a generic state  $|\psi\rangle$  can be written as:

$$|\psi\rangle = \sum_{s_1, s_2, \dots, s_N} c_{s_1 \dots s_N} |s_1 s_2 \dots s_N\rangle, \quad s_i = 0, 1, \dots, 5.$$

Therefore, to represent  $|\psi\rangle$ , it is necessary to specify  $d^N$  coefficients, namely  $c_{s_1 \dots s_N}$ . However, this number grows exponentially with  $N$ , making any direct simulation impractical even for moderate values of  $N$ .

To overcome this issue, we use *Tensor Networks* [12], which are a class of numerical techniques that allow the state  $|\psi\rangle$  to be represented in a compact form. A detailed explanation of their structure goes beyond the scope of this work.

Here we give a qualitative idea of their functioning, focusing on the concept of bond dimension and starting from the simplest case: a system composed of only two subsystems ( $N = 2$ ). In this case, the state can be written as:

$$|\psi\rangle = \sum_{i,j} M_{ij} |i\rangle \otimes |j\rangle,$$

where  $M_{ij}$  is the coefficient matrix, which can always be decomposed by a *Singular Value Decomposition (SVD)*:

$$M_{ij} = \sum_{k,k'} U_{ik} D_{kk'} V_{k'j} ,$$

where  $U$  and  $V$  are unitary matrices and  $D$  is a diagonal matrix with elements  $D_{kk} = \lambda_k$ . Substituting this expression into the expression for  $|\psi\rangle$ , we obtain the *Schmidt decomposition*:

$$|\psi\rangle = \sum_k \lambda_k |\alpha_k\rangle \otimes |\beta_k\rangle ,$$

with  $|\alpha_k\rangle = \sum_i U_{ik} |i\rangle$  and  $|\beta_k\rangle = \sum_j V_{jk}^* |j\rangle$ , which form orthonormal bases of the respective subspaces. In particular, the eigenvalues of  $D$  provide an indication of the entanglement between the two subsystems: if there is only one  $\lambda_i \neq 0$ , the system is separable; otherwise, it is entangled. From a physical point of view, we represent the information related to the individual subsystems in  $U$  and  $V$ , while their correlations are encoded in  $D$ .

In the case of a system with  $N > 2$  subsystems, this procedure is generalized by applying the decomposition to each pair of consecutive sites; thus, at each cut between two blocks of the system, a matrix  $D$  with its own eigenvalues is obtained. This specific decomposition leads to a particular class of tensor network called Matrix Product States (MPS).

**Example.** Consider a system composed by  $N = 3$  subsystems,  $A, B, C$ , and a pure state:

$$|\psi\rangle = \sum_{i,j,k} c_{ijk} |i\rangle_A \otimes |j\rangle_B \otimes |k\rangle_C .$$

**First cut:** we divide  $A$  from  $(BC)$ , rewriting  $c_{ijk} = M_{i,(jk)}$ , applying the SVD and obtaining the Schmidt decomposition:

$$M_{i,(jk)} = \sum_{\alpha} U_{i\alpha} D_{\alpha\alpha}^{[1]} V_{\alpha,(jk)} \quad |\psi\rangle = \sum_{\alpha} \lambda_{\alpha}^{[1]} |\alpha\rangle_A \otimes |\alpha'\rangle_{BC}, \quad \lambda_{\alpha}^{[1]} = D_{\alpha\alpha}^{[1]},$$

with  $|\alpha\rangle_A = \sum_i U_{i\alpha} |i\rangle_A$  e  $|\alpha'\rangle_{BC} = \sum_{j,k} V_{\alpha,(jk)} |j\rangle_B \otimes |k\rangle_C$ .

**Second cut:** we now decompose  $|\alpha'\rangle_{BC}$ , dividing the subsystems  $B, C$ . Defining  $W_{(\alpha j),k} := V_{\alpha,(jk)}$  we can apply the SVD a second time:

$$W_{(\alpha j),k} = \sum_{\beta} U'_{(\alpha j),\beta} D_{\beta\beta}^{[2]} V'_{\beta,k} \quad |\alpha'\rangle_{BC} = \sum_{\beta} \lambda_{\beta}^{[2]} |\beta(\alpha)\rangle_B \otimes |\beta'\rangle_C, \quad \lambda_{\beta}^{[2]} = D_{\beta\beta}^{[2]},$$

with  $|\beta(\alpha)\rangle_B = \sum_j U'_{(\alpha j),\beta} |j\rangle_B$  e  $|\beta'\rangle_C = \sum_k V'_{\beta,k} |k\rangle_C$ . Combining the two cuts we obtain:

$$|\psi\rangle = \sum_{\alpha,\beta} \lambda_{\alpha}^{[1]} \lambda_{\beta}^{[2]} |\alpha\rangle_A \otimes |\beta(\alpha)\rangle_B \otimes |\beta'\rangle_C .$$

In this way, the coefficients  $c_{ijk}$  can be rewritten as a product of matrices, i.e.  $D, U, V$ , leading to the MPS discussed above.

Now, assuming that, as occurs in many physical systems, the eigenvalues  $\lambda_k$  decay exponentially:

$$\lambda_k \sim e^{-\alpha k} ,$$

it is possible to reduce the number of  $\lambda_k$ , retaining only the first  $\chi$  most relevant terms, where  $\chi$  is called *bond dimension*. Essentially, we limit the complexity of the representation, neglecting minor correlations between subsystems. It can be shown that this approach allows a transition from a description of the state with  $d^N$  coefficients to an approximate description with a number of parameters that scales as  $\mathcal{O}(N \cdot d \cdot \chi^2)$ .

### 3.1.2 Time Discretization and $\Delta t$

To solve the Schrödinger equation, we use the *Time-Dependent Variational Principle* (TDVP) algorithm, whose detailed explanation lies beyond the scope of this thesis; here we provide only a qualitative description of its functioning. The idea of TDVP is not to consider the evolution in the whole Hilbert space, but rather only in the subspace consisting of states represented as MPS with a fixed bond dimension  $\chi$ .

Concretely, the time axis is discretized into intervals of size  $\Delta t$ , and at each time step the MPS is evolved by updating one tensor at a time. To perform this update, we construct an “effective Hamiltonian” by projecting the global Hamiltonian  $H$  onto the space of states represented as MPS and expressing it in terms of local Hamiltonians acting on each tensor of the MPS. The update is then implemented by computing the exponential of these local effective Hamiltonians and applying it to the corresponding tensor.

One of the main numerical error sources of this algorithm is the following: although the Hamiltonian  $H$  is constant, its projection onto the MPS manifold changes because the state itself changes after the evolution, therefore the effective Hamiltonian is time-dependent. Nevertheless, in TDVP the same effective Hamiltonian is used throughout the entire time step  $\Delta t$ , although in reality it changes. This leads to a numerical error that grows with  $\Delta t$ .

## 3.2 Observed Quantities in the Simulations

Once the simulation parameters have been established, we select a set of quantities to summarize the most relevant aspects of the system’s dynamics.

### 3.2.1 Local Quantities

In this section, we list three local quantities that have been observed during the simulations: pointer configuration, tape, and pointer position. Each of these quantities has been defined and computed according to the following scheme:

1. Representation of the states of a single qudit using basis vectors  $\{|j\rangle\}$  of  $\mathbb{C}^6$ :

$$0 \longrightarrow (1, 0, 0, 0, 0, 0) \quad 1 \longrightarrow (0, 1, 0, 0, 0, 0) \quad \dots \quad 5 \longrightarrow (0, 0, 0, 0, 0, 1)$$

2. Definition of a suitable diagonal operator  $\hat{O}$  acting on a qudit, i.e., a  $6 \times 6$  matrix such that  $\hat{O}|j\rangle = \lambda_j|j\rangle$ ;
3. Construction of  $N$  operators  $\hat{O}_i$  acting locally on the  $i$ -th cell via  $\hat{O}$ :

$$\hat{O}_i = \mathbb{I} \otimes \mathbb{I} \otimes \dots \otimes \hat{O} \otimes \dots \otimes \mathbb{I};$$

4. Computation of the  $N$  expectation values of  $\hat{O}_i$  on the state of the system at time  $t$ :

$$\langle \psi(t) | \hat{O}_i | \psi(t) \rangle;$$

5. Color plot in which the horizontal axis represents time  $t$ , the vertical axis represents the  $N$  cells, and the color at each point  $(t, i)$  is proportional to the expectation value  $\langle \hat{O}_i \rangle$ .

Once the method for constructing and computing the local quantities is specified, we can now define the corresponding operator  $\hat{O}$  for each observable, justifying the choice.

**Pointer configuration.** The first quantity we consider is the pointer configuration, defined through  $C$ :

$$C = \text{diag}(0, 0, -1, -1, 1, 1),$$

which, when applied to a basis state of  $\mathbb{C}^6$ , adds a factor equal to the value of the pointer if present or 0 if absent. In this way, it distinguishes the cells currently undergoing an update from the others. This observable provides a map of the average logical value associated with the pointer: in the classical case, at each time step,  $\langle C_i \rangle$  would be  $\pm 1$  for a single cell and zero elsewhere, whereas now it can take intermediate values in  $[-1, 1]$  and be non-zero at multiple cells simultaneously.

**Tape.** Another quantity we want to study is the *tape*, defined through  $T$ :

$$T = \text{diag}(1, -1, 1, -1, 1, -1),$$

which, when applied to a basis state of  $\mathbb{C}^6$ , adds a factor equal to the negative of the logical value of the cell,  $s_i$ , regardless of the presence of the pointer. This allows us to analyze how the logical content of the tape evolves over time.

**Pointer position.** Finally, we consider the *pointer position*, defined through  $P$ :

$$P = \text{diag}(0, 0, 1, 1, 1, 1),$$

which is a local operator that identifies the presence of the pointer. Unlike the pointer configuration, which assigns a factor  $\pm 1$  depending on the pointer's logical value, the position observable simply returns 1 in all cases where the pointer is present. This quantity is useful to build the probability distribution of the pointer's position along the tape and to compute its average position.

Index	$(p_i, s_i)$	$C$	$T$	$P$
0	(0, -1)	0	1	0
1	(0, 1)	0	-1	0
2	(-1, -1)	-1	1	1
3	(-1, 1)	-1	-1	1
4	(1, -1)	1	1	1
5	(1, 1)	1	-1	1

Table 3.1: Eigenvalues associated to each basis state of the qudit, after applying  $C$ ,  $T$ , or  $P$ .

### 3.2.2 IPR

Another quantity we consider is the *Inverse Participation Ratio* (IPR), which quantifies the localization of a quantum state with respect to a given basis of the Hilbert space. Here, we consider the localization with respect to the computational basis. Specifically, given a normalized state  $|\psi\rangle$ , expressed in the computational basis  $\{|i\rangle\}$ , the IPR is defined as:

$$\text{IPR} = \sum_i |\langle i|\psi\rangle|^4, \quad |\langle i|\psi\rangle| < 1$$

This indicator quantifies how much the wave function is distributed among the different configurations of the basis: if  $\text{IPR} \sim 1$ , the state is strongly localized, meaning that the probability of finding the system in a small number of configurations is high; conversely, if  $\text{IPR} \ll 1$ , the state is delocalized, i.e., it is spread across many configurations.

In our case, the IPR allows us to analyze whether, during the evolution, the initial state tends to spread across the whole diagram or remains confined to a particular region. In the simulations, the IPR is recorded at each time step, providing a direct measure of the evolution of localization over time.

### 3.2.3 Maximum Bond Entropy

The *Maximum Bond Entropy* provides a measure of the entanglement generated in the system during the evolution. Specifically, to calculate it, we divide the system into two subsystems  $A$  and  $B$ , through a cut between the central cell and its neighbor. Given the global state of the system  $|\psi\rangle \in (\mathbb{C}^6)^{\otimes N}$ , we compute the reduced density matrix  $\rho_A$  of the subsystem  $A$ :

$$\rho_A = \text{Tr}_B (|\psi\rangle\langle\psi|).$$

Since we are dealing with pure states, we can then measure the entanglement between the two subsystems through the von Neumann entropy  $S_N$ :

$$S_N(\rho_A) = -\text{Tr}(\rho_A \log_2 \rho_A).$$

Here,  $\log_2 \rho_A$  is computed by diagonalizing  $\rho_A$  and taking the logarithm of the individual eigenvalues. In particular, in the tensor network formalism,  $\rho_A$  is already diagonalized via the Schmidt decomposition of the state:

$$|\psi\rangle = \sum_{\alpha} \lambda_{\alpha} |\phi_{\alpha}^A\rangle \otimes |\phi_{\alpha}^B\rangle, \quad \rho_A = \sum_{\alpha} \lambda_{\alpha}^2 |\phi_{\alpha}^A\rangle\langle\phi_{\alpha}^A|$$

where  $\{|\phi_{\alpha}^{A(B)}\rangle\}$  are orthonormal bases of the respective subspaces. The von Neumann entropy then becomes:

$$S_N = -\sum_{\alpha} \lambda_{\alpha}^2 \log_2 \lambda_{\alpha}^2.$$

Finally, it can be shown that  $S_N < \log_2 m$ , where  $m$  is the number of eigenvalues  $\lambda_{\alpha}$  which scales exponentially with the size of the system  $N$ .

In conclusion, this quantity measures the entanglement generated between the two halves of the system, and it is expected to grow over time until it reaches a stationary regime determined by the system size or by the bond dimension  $\chi$ , since:

- The local rules of the automaton progressively generate correlations between neighboring cells, and these correlations propagate along the tape, increasing the entanglement between the left and right halves of the chain, up to a maximum value of the order of the system size  $S_N = \log_2 m \propto N$ .
- The use of tensor networks involves keeping only the first  $\chi$  values of  $\lambda_{\alpha}$ , i.e., it artificially reduces the entanglement. This phenomenon may lead to saturation before the actual maximum value is achieved. In particular, the maximum entanglement reachable with a bond dimension  $\chi$  corresponds to the maximally mixed state and is  $\log_2 \chi$ .
- If the presence of  $\chi$  leads to a significant reduction of the entanglement, the truncated eigenvalues are not really negligible, and therefore it is necessary to increase the bond dimension to obtain more reliable results.

### 3.2.4 Current

The last quantity we want to study is the *current*, namely the flow of the pointer along the tape from left to right, independently of the tape content and of the internal state of the pointer itself. The current is constructed in analogy with tight-binding models, where it describes the transport of particles between adjacent cells and it is proportional to the imaginary part of the hopping operators that *create* the particle at one cell and *annihilate* it at another.

**Derivation of the current.** A simplified way to obtain the formula for the current in a tight-binding model is the following. We define a Hamiltonian  $H$  in which particles can only hop between neighboring sites:

$$H = \sum_i (C_{i+1}^{\dagger} C_i + C_i^{\dagger} C_{i+1}), \quad n_i = C_i^{\dagger} C_i,$$

where  $C_i^\dagger, C_i$  are respectively the creation and annihilation operators at site  $i$ , while  $n_i$  is the number operator that counts how many particles are at site  $i$ . Then, substituting  $H$  into the Heisenberg equation for  $n_i$ , we obtain:

$$\frac{d}{dt}n_i = i[H, n_i] = i \left( C_{i-1}^\dagger C_i - C_i^\dagger C_{i-1} \right) - i \left( C_{i+1}^\dagger C_i - C_i^\dagger C_{i+1} \right) := J_{i-1 \rightarrow i} - J_{i \rightarrow i+1}.$$

In which we have defined the current from site  $i$  to site  $i + 1$  as:

$$J_{i \rightarrow i+1} = i \left( C_{i+1}^\dagger C_i - C_i^\dagger C_{i+1} \right).$$

This form corresponds to the imaginary part of the hopping terms, and it is a natural definition of the current because it guarantees the validity of the continuity equation and the conservation of the number of particles. Finally, we note that our model satisfies the conservation of the number of pointers, and its Hamiltonian is also composed of hopping terms, so the analogy with tight-binding models is justified.

**Current of the pointer.** To define the current of the pointer we consider creation and annihilation operators,  $C_{\pm, i}^\dagger$  and  $C_{\pm, i}$ , which respectively introduce or remove a pointer with internal state  $\pm 1$  at cell  $i$ . The local current from cell  $i$  to cell  $i + 1$  is defined as the sum of all possible processes in which a pointer is destroyed at one cell and created at the next one, assuming that in the process the pointer may also change its internal state:

$$J_i = i \left( C_{-, i} C_{+, i+1}^\dagger + C_{+, i} C_{-, i+1}^\dagger + C_{-, i} C_{-, i+1}^\dagger + C_{+, i} C_{+, i+1}^\dagger - h.c. \right).$$

The total current is then obtained as the sum over all cells:

$$J = \sum_i J_i.$$

**Creation and annihilation operators.** To explicitly construct the creation and annihilation operators  $C_{\pm, i}^\dagger, C_{\pm, i}$  we observe that they must encode on cell  $i$  the following transitions:

- $C_{-, i}^\dagger$ , creation of a pointer with  $p_i = -1$ :

$$0 \rightarrow 2 \quad 1 \rightarrow 3 \quad \text{that is} \quad (0, -1) \rightarrow (-1, -1) \quad (0, 1) \rightarrow (-1, 1)$$

- $C_{-, i}$ , annihilation of a pointer with  $p_i = -1$ :

$$2 \rightarrow 0 \quad 3 \rightarrow 1 \quad \text{that is} \quad (-1, -1) \rightarrow (0, -1) \quad (-1, 1) \rightarrow (0, 1)$$

- $C_{+, i}^\dagger$ , creation of a pointer with  $p_i = 1$ :

$$0 \rightarrow 4 \quad 1 \rightarrow 5 \quad \text{that is} \quad (0, -1) \rightarrow (1, -1) \quad (0, 1) \rightarrow (1, 1)$$

- $C_{+, i}$ , annihilation of a pointer with  $p_i = 1$ :

$$4 \rightarrow 0 \quad 5 \rightarrow 1 \quad \text{that is} \quad (1, -1) \rightarrow (0, -1) \quad (1, 1) \rightarrow (0, 1)$$

Identifying, as done previously, each state of the qudit  $i$  with a basis vector of  $\mathbb{C}^6$ , these operators can be written as the following tensor product:

$$C_{\pm, i} = \mathbb{I} \otimes \cdots \otimes (c_{\pm})_i \otimes \cdots \mathbb{I}$$

where  $c_{\pm, i}$  are  $6 \times 6$  matrices encoding the transitions listed above. As an example we report the case of  $C_{-, 1}^\dagger$ .

**Example.** Consider the case of  $C_{-,1}^\dagger$ . The  $6 \times 6$  matrix  $c_-$  must encode the transitions  $0 \rightarrow 2$ ,  $1 \rightarrow 3$ , which, in the representation through basis vectors of  $\mathbb{C}^6$ , correspond to the transitions:

$$(1, 0, 0, 0, 0, 0) \rightarrow (0, 0, 1, 0, 0, 0) \quad (0, 1, 0, 0, 0, 0) \rightarrow (0, 0, 0, 1, 0, 0)$$

Therefore the matrix  $c_-$  is:

$$c_- = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and consequently:

$$C_{-,1} = c_- \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I}$$

**Interpretation of the current.** The sign and the value of the current provide information about the direction and the average velocity of propagation of the pointer:

- $\langle J \rangle > 0$ : average motion to the right,
- $\langle J \rangle < 0$ : average motion to the left,
- $\langle J \rangle \approx 0$ : balanced motion or absence of net propagation.

In superposition states, intermediate values indicate the coexistence of propagation in both directions. Moreover, since the current measures the net probability flux of the pointer, its value is directly proportional to the pointer's *average velocity* of propagation.

### 3.3 Initial conditions

Once the automaton's classical configurations are represented as computational-basis states, the choice of initial conditions becomes more flexible. In particular, in the simulations we use two initial conditions: the first one, similar to the classical system, is a single node of the diagram; the second one is a wave packet, i.e., a superposition of basis states associated with nodes connected in the graph.

#### 3.3.1 Computational-basis state

We start the system evolution from an initial condition similar to the classical one, namely from a single node of the diagram. In other words, we set the system initial state  $|\psi_0\rangle$  equal to one of the computational-basis states. In particular, we choose the same state that generates the Sierpiński triangles shown in Fig. 1.6 in Chapter 1, i.e., a state in which the tape has all cells initialized to  $-1$  and the central one to  $+1$ . We also recall that in the classical automaton the boundary cells, which have only one neighbor to their right/left, have a *virtual cell* in state  $-1$  on their left/right. Therefore, using the extended alphabet of the SRS, the initial state becomes:

$$|\psi_0\rangle = |200 \dots 1 \dots 00\rangle$$

Indeed:

- the tape has the central cell in state  $+1$  and all the others in state  $-1$ ;
- the pointer is present only on the first cell: as discussed in Chapter 1, this type of configuration corresponds to a node of the classical automaton's diagram;
- the pointer is initialized in state  $-1$  to respect the boundary conditions of the classical automaton: since  $p_i = s_{i-1}$ , the first cell has a *virtual cell* to its left in state  $-1$ .

### 3.3.2 Wave packet

The second initial condition from which we chose to start is a Gaussian wave packet that moves on the diagram, constructed as follows:

- start from the classical string  $s_0 = 200 \dots 1 \dots 00$  defined in the previous section;
- apply the local rule in the SRS formalism iteratively to  $s_0$ , sliding the pointer until it reaches the end of the tape: this generates the ordered sequence of  $N$  configurations  $\{s_j\}_{j=0, \dots, N-1}$  that connects the node  $s_0 = 200 \dots 1 \dots 00$  to the next one;
- associate to each  $s_j$  the *position*  $j$  and build the initial state as a superposition of the states  $\{|s_j\rangle\}$ :

$$|\psi(0)\rangle = \frac{1}{\mathcal{N}} \sum_{j=0}^{L-1} e^{ikj} \exp\left(-\frac{(j-\mu)^2}{2\sigma^2}\right) |s_j\rangle,$$

where:

$$k = \pi/2, \quad \mu = \frac{N-1}{2}, \quad \sigma = \frac{N}{6}.$$

In particular,  $\mathcal{N}$  is the normalization constant, and  $k = \pi/2$  is the mean momentum introduced to produce a non-stationary packet that propagates in a definite direction at maximal group velocity. Finally,  $\mu$  and  $\sigma$  are chosen so that the packet is well localized with negligible tails, with its peak at the *center* of the arc that connects two configurations of the classical automaton diagram.

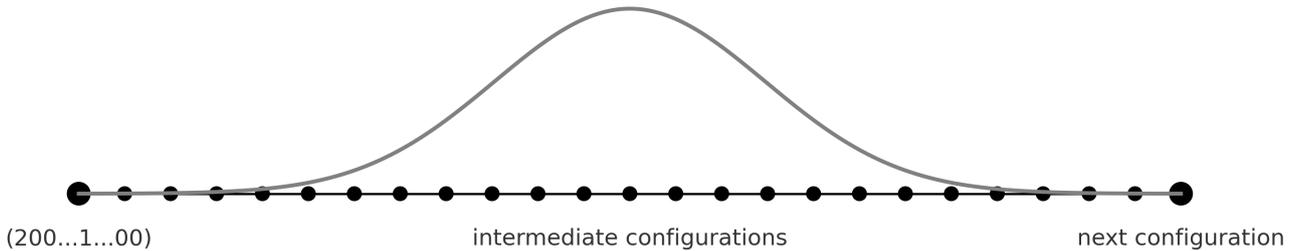


Figure 3.1: Representation of the initial wave packet for a system with 25 cells.

## 3.4 Simulation results

In this section, we present the results of the simulations for the two different initial conditions: the computational basis state and the wave packet. For each of the two initial conditions, we compare the IPR, the Max B.E., and the Current of simulations with different parameters  $\chi$  and  $\Delta t$ , in order to study their effect on the simulations themselves. In particular, since the error due to  $\Delta t$  accumulates over time, we expect the simulations to diverge at large  $t$ . The plots of IPR, Max B.E., and Current also allow us to characterize the degree of localization of the system and the growth of quantum correlations. Finally, we compare the local quantities, studying their transport and distribution along the cells and highlighting the differences between the two initial conditions.

We choose to simulate a Class 3 automaton, namely Rule 90, because, as discussed in Chapter 1, unlike the automata of the other classes, it exhibits chaotic and non-trivial pseudorandom behaviors (such as the formation of Sierpinski triangles), showing fractal structures and the presence of large attractors in the state-transition diagrams. Moreover, we choose a time step  $\Delta t \sim 10^{-2}-10^{-3}$  and a bond dimension  $\chi$  in the range  $10^1-10^2$ , in order to ensure computationally feasible simulations, while avoiding too coarse approximations. Finally, we set the system size to  $N = 25$ , which is large

enough to capture the emergence of complex structures, but also sufficiently small to allow numerical simulations. Summarizing, we set:

$$\chi \sim 10^1\text{--}10^2, \quad \Delta t \sim 10^{-2}\text{--}10^{-3}, \quad N = 25.$$

### 3.4.1 Computational basis state

Here we report the plots of IPR, Max B.E., and Current of the simulations that reproduce the evolution of the basis state, for different values of  $\chi$  and  $\Delta t$ <sup>1</sup>.

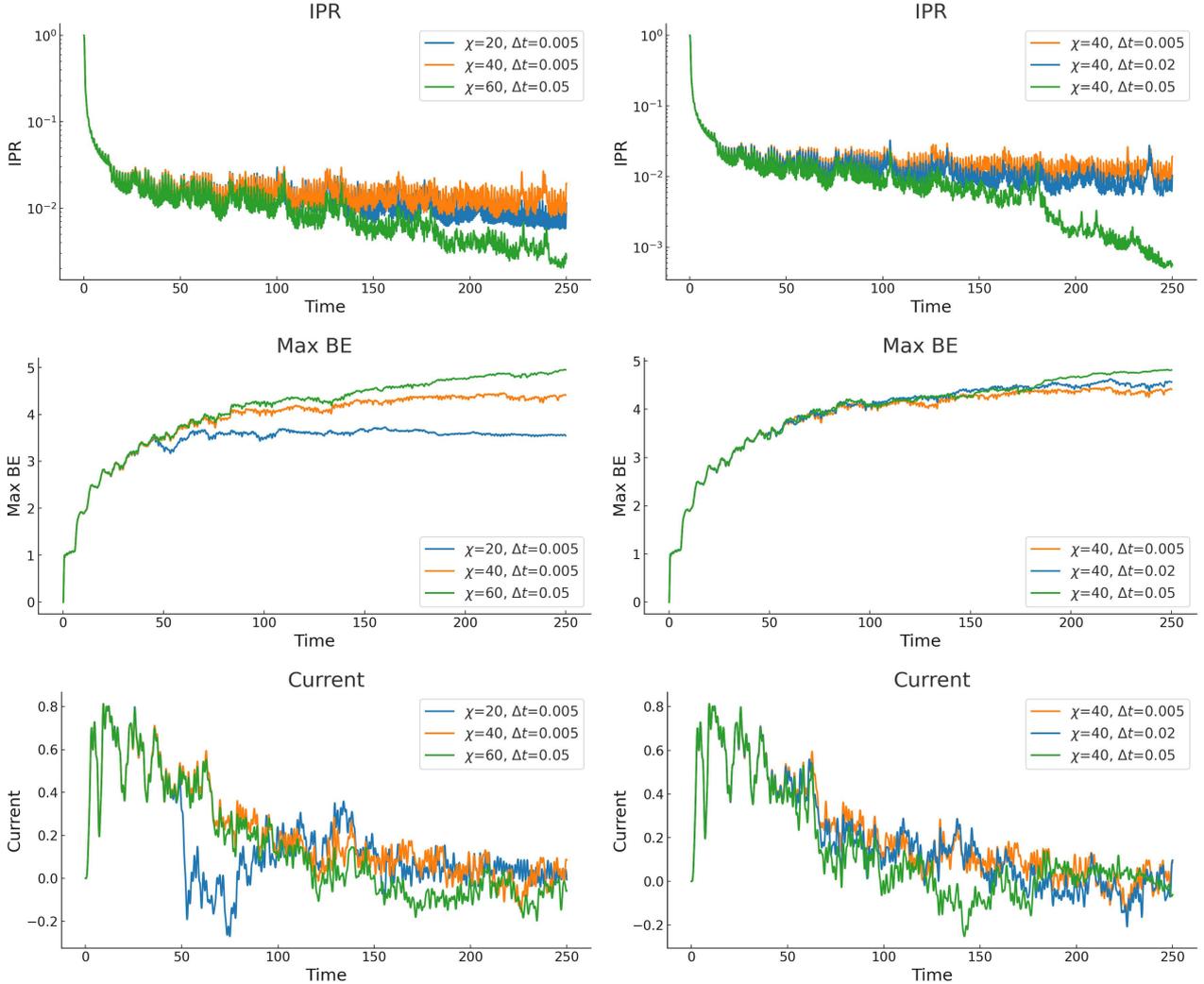


Figure 3.2: Comparison plots of IPR, Max B.E., and current for different parameters with the computational basis state as the initial condition. On the left, the effect of varying  $\chi$ ; on the right, the effect of varying  $\Delta t$ .

The plots show that:

- At  $t = 0$ ,  $\text{IPR} = 1$ , but it quickly drops to  $\text{IPR} \sim 10^{-2}$ , without significant differences among the plots with different parameters: this means that the system starts from a localized basis state, our initial condition, and becomes quickly delocalized over the diagram, evolving into a superposition of basis states.
- The Max B.E. saturates rapidly, that is, the system quickly becomes strongly correlated, and this correlation increases with  $\chi$ : this phenomenon may be related to the fact that lower values of  $\chi$  artificially reduce the entanglement of the system. In contrast, changes in  $\Delta t$  do not seem to

<sup>1</sup>In the case  $\chi = 60$  the only available simulation is the one with  $\Delta t = 0.05$ , although it would have been more appropriate  $\Delta t = 0.005$ .

affect this quantity. However, in all cases, the saturation occurs before the expected theoretical maximum value, namely  $\log_2 \chi$ .

- The current initially has a preferred direction (to the right), but for  $t \gtrsim 50$  it settles to 0, and this effect is related to the delocalization discussed above: once all points of the graph are reached after a short time, motion can proceed in any direction on the graph, i.e., the pointer can move to the right as well as to the left, so its net transport is null.

In light of these observations, we expect the pointer position and configuration to be quickly distributed along all cells, i.e., the probability of finding the pointer in a given state on each cell becomes more and more similar for  $t \gtrsim 50$ .

### 3.4.2 Wave packet

Here we report the plots of IPR, Max B.E., and Current of the simulations that reproduce the evolution of the wave packet, for different values of  $\chi$  and  $\Delta t$ .

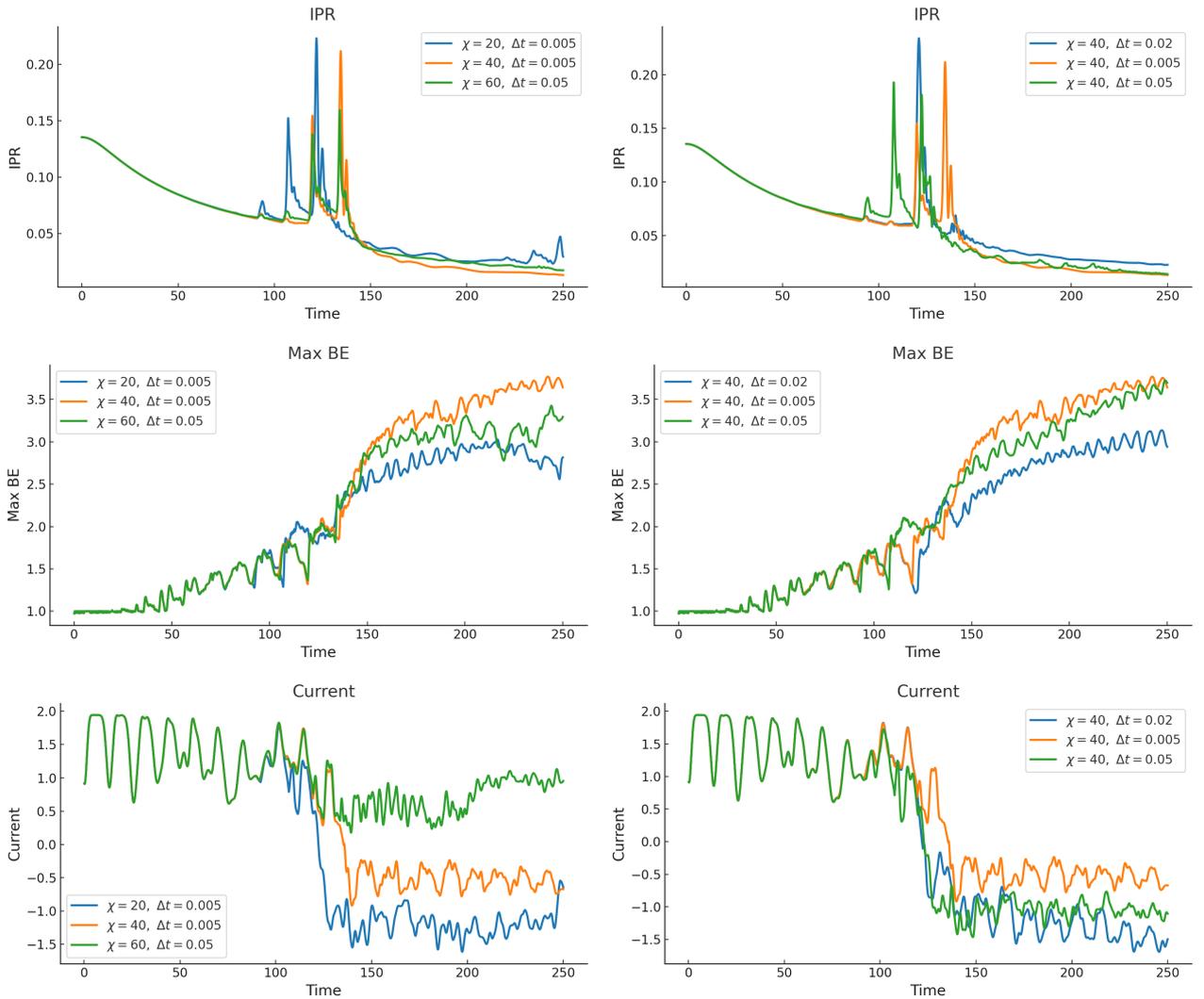


Figure 3.3: Comparison plots of IPR, Max B.E., and current for different parameters for the wave packet. On the left, the effect of varying  $\chi$ ; on the right, the effect of varying  $\Delta t$ .

For the wave packet, we observe different behaviors from the previous case:

- The IPR decreases much more slowly than before and is always  $\text{IPR} \sim 10^{-1}$ , that is, the packet is much less delocalized. Moreover, no significant differences are observed when changing the parameters, except for the presence of a peak that occurs at  $t \sim 140$  for  $\chi = 40$ .

- The Max B.E., that is, the system entanglement, grows slowly without saturating and is lower than in the previous case. In addition, in this case, Max B.E. also appears to be influenced by changes in  $\Delta t$ .
- The current initially has a well-defined direction to the right, consistent with the fact that the packet has the maximum group velocity in that direction. However, at  $t \sim 140$ , i.e., when the IPR exhibits the peak, the current changes sign, suggesting a reflection. This phenomenon may be due to the fact that at  $t \sim 140$  the boundary cells of the tape are also updated, as we will discuss in the next section. Finally, we note that for  $\chi = 60$  the current decreases less, but this may be an effect due to the large  $\Delta t$ , whose error accumulates over time.

In this case, the dynamics is therefore that of a *quantum walk* in which the packet propagates on the graph.

### 3.4.3 Comparison of local quantities

Finally, we report the plots of the local quantities (pointer configuration and position, tape) for both initial conditions, with parameters  $\chi = 40$ ,  $\Delta t = 0.005$ .

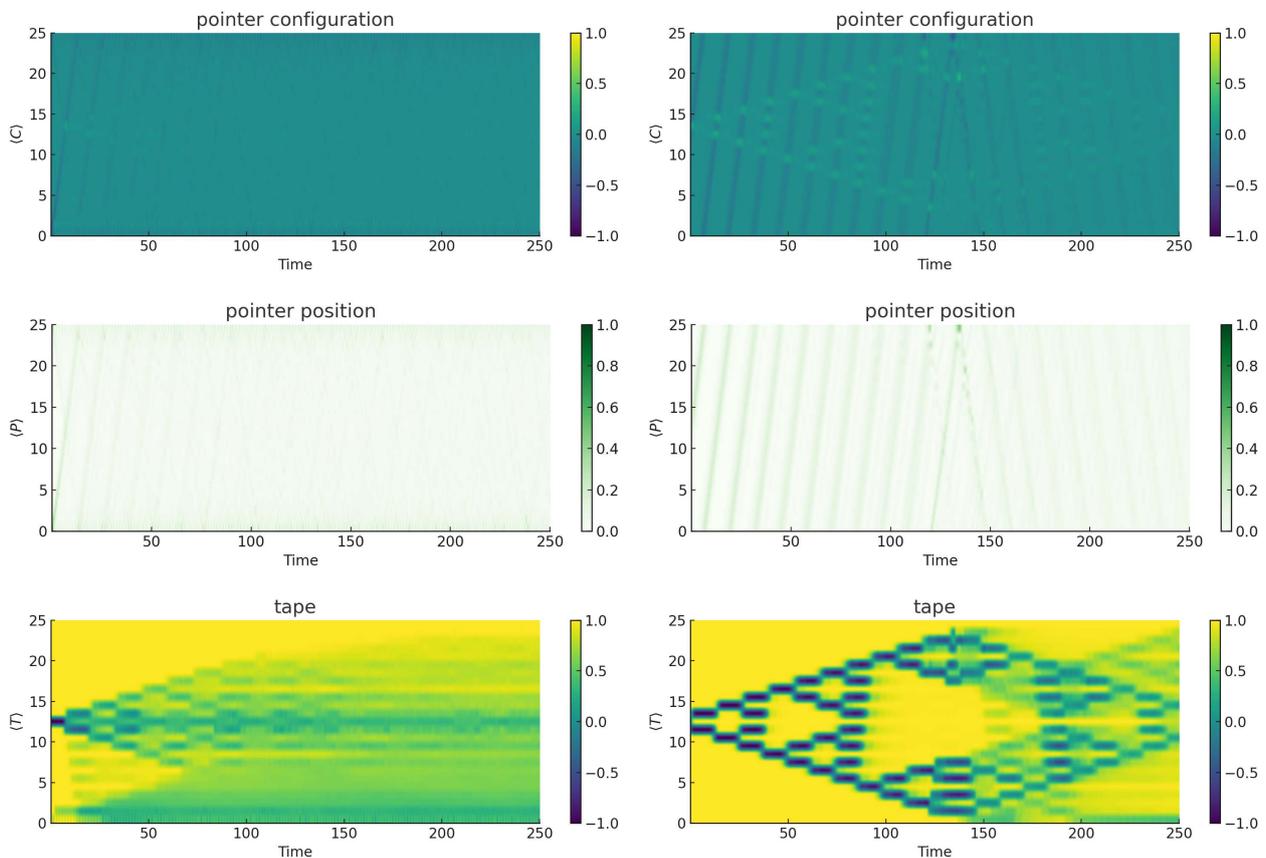


Figure 3.4: Comparison of local observables for the two initial conditions. On the left, the computational basis state; on the right, the wave packet.

From the plots, we observe that:

- The pointer configuration and position for the computational basis state are well defined only at the beginning: this is consistent with  $\text{IPR} \sim 10^{-2}$ , i.e., a delocalized state, and almost zero current. In other words, it becomes impossible to predict which position or configuration is most likely to occur at long times.
- The pointer configuration and position for the wave packet, instead, are more defined since the delocalization is smaller. Moreover, at  $t \sim 140$  the pointer seems to undergo a reflection, which

corresponds to the sign change of the current seen above and to the IPR peak. It is interesting to note that this phenomenon occurs when the tape edge cells change sign and therefore do not depend on the pointer, but on the automaton cells. We also observe that the reflection is not complete, but there is a small transmitted component: consequently, the lines of pointer position and configuration are less sharp, and interference and superposition effects occur.

- In the case of the wave packet, the tape exhibits a well-defined pattern that reproduces the Sierpinski triangles up to  $t \sim 140$ ; however, following the reflection and partial transmission of the pointer, this pattern becomes less intense (i.e., less probable) and a mirror pattern due to the reversed current overlaps it. This phenomenon distinguishes this plot from the classical case and shows the effect of quantum superposition.
- In the case of the computational basis state, although the tape starts from a well-defined state, it shows Sierpinski triangles that are less sharp and soon become indistinguishable, since the state evolves into a superposition of states in which all configurations are likely to occur.
- There is an asymmetry in the tape plot when starting from a the computational basis state: this is related to the pointer initial position, i.e. the first cell, and is consistent with the observation that the current is positive at early times (on average, the pointer moves to the right). In contrast, in the case of the wave-packet, the pattern is symmetric, since the packet is centered on the state with the pointer located on the central cell, i.e. halfway between an automaton configuration and the its updated one (see Fig. 3.1).

In general, we observe how the choice of initial condition determines two well-distinct dynamical regimes. In the case of the wave packet, there is a smaller delocalization and the propagation exhibits intensity modulations and interference that confirm the genuinely quantum component of the dynamics. Instead, the computational basis state becomes quickly delocalized and is characterized by a less directional pointer dynamics and higher entanglement.

# Conclusion

In this thesis, we have shown how, starting from the simple rules of an elementary cellular automaton, it is possible to construct a quantum system that exhibits complex emergent phenomena and we highlighted the system's behavior in terms of localization and entanglement.

To achieve this, we first introduced the equivalence between the automaton dynamics and the state-transition diagram, and subsequently encoded the classical configurations of the automaton into computational basis states of a quantum many-body system. With the introduction of an additional degree of freedom, namely the pointer, and the string rewriting formalism, we built a local Hamiltonian  $H$  that directly encodes the automaton evolution rules and their order of application, allowing the description of the quantum dynamics as a *quantum walk* on an extended diagram that simulates the original one.

Finally, the use of *Tensor Networks* allowed us to analyze the growth of correlations and localization mechanisms, reducing at the same time the computational cost of simulations.

The analysis of global quantities such as the IPR, the maximum bond entropy, and the current, and local quantities, allowed us to characterize the system dynamics starting from two different initial conditions: the computational basis state and the wave packet. In the first case, we observed rapid delocalization and a growth of correlations in the system, whereas in the second case these two phenomena were slower and more limited. Moreover, the wave packet propagating in one direction effectively simulated the irreversible evolution of the original automaton, but also showed the effects of quantum superposition.

This work is part of a line of research that aims to connect classical and quantum systems, offering new tools to understand emergent phenomena and complexity in quantum systems. Future steps may include extending the study to other cellular automata, introducing disorder or noise into the model, and exploring the possibility of implementing such dynamics on quantum computers.

# Bibliography

- [1] Stephen Wolfram. “Cellular automata as models of complexity.” In: *Nature* 311 (1984), pp. 419–424.
- [2] Stephen Coombes. “The Geometry and Pigmentation of Seashells.” Department of Mathematical Sciences, University of Nottingham, Nottingham, UK. 2009.
- [3] Yves Bouligand. “Fibroblasts, Morphogenesis and Cellular Automata.” In: *Disordered Systems and Biological Organization*. Ed. by E. Bienenstock, F. Fogelman, and G. Weisbuch. Springer, 1986.
- [4] Terry Farrelly. “A review of Quantum Cellular Automata.” In: *Quantum* 4 (2020), p. 368. DOI: 10.22331/q-2020-11-30-368.
- [5] P. Arrighi. “An overview of quantum cellular automata.” In: *Natural Computing* 18 (2019), pp. 885–899.
- [6] Davide Rattacaso et al. *Quantum algorithms for equational reasoning*. 2025. arXiv: 2508.21122 [quant-ph]. URL: <https://arxiv.org/abs/2508.21122>.
- [7] John Preskill. “Quantum Computing in the NISQ era and beyond.” In: *PRX Quantum* 2.1 (2021), p. 017003. DOI: 10.1103/PRXQuantum.2.017003.
- [8] Wolfram Demonstrations Project. *Cellular Automaton State Transition Diagrams*. 2025. URL: <https://demonstrations.wolfram.com/CellularAutomatonStateTransitionDiagrams/>.
- [9] Wikipedia contributors. *Cellular automaton — Wikipedia, The Free Encyclopedia*. 2025. URL: [https://en.wikipedia.org/wiki/Cellular\\_automaton](https://en.wikipedia.org/wiki/Cellular_automaton) (visited on 08/04/2025).
- [10] Salvador E. Venegas-Andraca. “Quantum walks: a comprehensive review.” In: *Quantum Information Processing* 11.5 (2012), pp. 1015–1106.
- [11] Flavio Baccari et al. *Quantum TEA: qtealeaves*. Feb. 1, 2025. DOI: 10.5281/zenodo.10498928. URL: <https://doi.org/10.5281/zenodo.10498928>.
- [12] Mari Carmen Bañuls. “Tensor Network Algorithms: A Route Map.” In: *Annual Review of Condensed Matter Physics* 14 (2023), pp. 173–191. DOI: 10.1146/annurev-conmatphys-040721-022705.
- [13] Edward Farhi and Sam Gutmann. “Quantum computation and decision trees.” In: *Physical Review A* 58 (1998), pp. 915–928.
- [14] Jutho Haegeman et al. “Unifying time evolution and optimization with matrix product states.” In: *Phys. Rev. B* 94 (16 Oct. 2016), p. 165116. DOI: 10.1103/PhysRevB.94.165116. URL: <https://link.aps.org/doi/10.1103/PhysRevB.94.165116>.