2 Chapter 2: The Logic of Self-Replication

We now turn to the problem sketched in the introduction: the design of a physical system capable of building an exact copy of itself. In this chapter, we present John von Neumann's solution to this challenge. His key insight is to separate the self-replicating system into two components: a machine, endowed with functional and constructive capabilities, and a tape (or blueprint) that contains the machine's description. Crucially, the tape plays a dual role: it is first interpreted as a set of instructions guiding the construction of a new machine, and then copied as uninterpreted data and inserted into the offspring. This separation resolves the apparent paradox that a machine must be more complex than what it produces.

Remarkably, this logical organization of self-replication is essentially the one later uncovered by molecular biology: genetic material is both read to build the organism and copied to transmit heredity. Von Neumann arrived at this architecture several years before the discovery of the structure of DNA.

In order to make these ideas mathematically precise, von Neumann worked in a highly abstract model of the physical world. Therefore, his construction cannot be straightforwardly translated into a real-world self-replicator; this remains a deeply intriguing open problem. However, his solution points to the fundamental principles underlying self-reproduction, that extend far beyond the specific model he used.

Von Neumann began developing the theory of self-reproducing automata in the late 1940s and first presented his ideas at the Hixon Symposium at Caltech in 1948 [7]. In his early work, he explored a kinematic model of self-replication. Although this model did not allow for a solution with full mathematical rigour, it already contained the core of his groundbreaking ideas, while avoiding the technical difficulties required for a fully formal treatment. He later formulated a rigorous solution within a more abstract framework known as a cellular automaton. We will follow this organization: first we explain the core ideas in the context of the kinematic model, and then we proceed with a detailed implementation in the cellular model.

2.1 Von Neumann's Kinematic Model

In his lectures at the University of Illinois, transcribed and completed in [9, Part I], von Neumann argued that, after a number of simplifications, a meaningful analogy can be drawn between living and computational systems; he referred to the latter broadly as automata. Driven by this analogy, he posed the question whether an automaton could replicate itself.

He pointed out that Turing's definition is not well suited to study self-replication in this sense: the output of a Turing machine is a sequence of symbols on a tape, not another Turing machine. In particular, the machine and the medium it acts upon are fundamentally different types of objects. As von Neumann stated in his lecture: "In all cases, the medium which is fed to the automaton and which is produced by the automaton is completely different from the automaton. In fact, the automaton doesn't produce any medium at all; it merely modifies a medium which is completely different from it."

The crucial way to overcome this limitation is to consider what we might call an *embodied Turing machine*: a physical substrate in which the Turing machine and its tape coexist and are constructed from the same materials.

The initial model he proposed for the design of embodied Turing machines is known as the *kinematic model.* It consists of a continuous space populated by free-floating particles: these serve as elementary building blocks for both the machine and the tape. Some of these particles represent matter, and ways to fuse or cut such building blocks. Others represent information processing:

these are blocks transmitting signals and implementing basic logic gates. The idea is that such elements can be further assembled into larger structures with various functionality, such as the tape consisting of a line of rigid symbols, a tape reader, or a sensing unit capable of determining the type of element it is in contact with.

Though this is not important, for concreteness we illustrate the kinds of elementary particles von Neumann was considering:

- **Information processing:** A *stimulus element* which receives and transmits stimuli, an *OR element* that acts as an OR gate of incoming signals, an *AND element*, a *NAND element*, and a *producer* – an element generating signals.
- **Matter organization:** A *rigid element* that does not transmit signals and may be connected to any other element. These connections are made by a *fusing element* and broken by a *cutting element*. The last, eighth part is a *muscle* which contracts as long as it is stimulated and enables movement.

Exercise 2.1. *Show that the NAND element alone suffices to implement any Boolean function, thereby reducing the number of elementary components from eight to four. Why do you think von Neumann nevertheless chose to include the other logical gates?*

2.1.1 The Core Idea of Self-Replication

We now make a strong assumption, which is exactly where a substantial amount of technical details are omitted: we assume that the available elements can be arranged so as to construct an arbitrary Turing machine operating on a tape composed of rigid symbols. We further assume we can extend this machine with a construction unit capable of assembling freely floating elements, available in unlimited supply, allowing it to build new machines according to the description encoded on the tape.

As in Turing's original construction, each machine can be assigned a canonical description in the form of a binary sequence. On this basis, von Neumann argues for the existence of a universal constructor $\mathcal{B}$: a machine which, when provided with the description of another machine $\mathcal{X}$, reads the tape and constructs $\mathcal{X}$ from the surrounding elements. He notes that the existence of $\mathcal{B}$ "is not qualitatively different from the type of argumentation with which Turing defined his universal automaton."

In addition, von Neumann posits the existence of a machine $\mathcal{A}$ that produces an exact copy of the tape it reads. This is not a far-fetched assumption, as the tape is a linear object, and can be read and copied sequentially in a straightforward way. Combining these, we define a machine $\mathcal{R}$ formed by composing $\mathcal{B}$ and $\mathcal{A}$; when $\mathcal{R}$ receives the description of a machine $\mathcal{X}$, first, $\mathcal{R}$ triggers the function of $\mathcal{B}$, which builds $\mathcal{X}$ (without any tape). Then, it triggers the activity of $\mathcal{A}$, which creates an exact copy of the tape. Finally, this new tape is inserted into the new machine.

Crucially, when $\mathcal{R}$ is supplied with a tape containing its own description, it first activates $\mathcal{B}$, which builds a copy of $\mathcal{R}$, initially without a tape. Then $\mathcal{A}$ is activated to copy the original tape and insert it into the newly constructed machine. In this way, a complete copy of $\mathcal{R}$, ready to self-replicate again, is produced.

Remark 2.2. *Notice the loose analogy between the above construction and the universal quine construction of Proposition* **??**. *In both cases, the seeming paradox of self-reference is resolved by decomposing the self-referential entity into two parts: $\mathcal{B}$, and $\mathcal{A}$. In both cases, $\mathcal{B}$ is a object with a rich functionality: in the case of quines, from the content of the tape $\mathcal{B}$ retrieves the code of the print machine that produced it, and prints this code. In von Neumann's case, from the content* *of the tape $\mathcal{B}$ retrieves the description of the machine and builds it from surrounding materials. Further, in both cases $\mathcal{A}$ plays the role of a data printer.*

In his outline, von Neumann introduced a revolutionary idea: the dual role of the tape's content: first as instructions to be interpreted for the construction of a new machine (translation), and then as data to be copied and inserted into that machine (transcription). Remarkably, this conceptual separation predates the discovery of DNA's structure; we discuss this striking historical parallel in Section 2.1.2.

2.1.2 Historical Window: Self-Replicating Machines and the Structure of DNA

We encourage the reader to listen to a remarkable interview with Sydney Brenner, Nobel laureate for his work on the genetic code and his foundational contributions to molecular biology, in which he reflects on the fascinating historical interplay between von Neumann's ideas on self-replication and the discovery of the structure of DNA. The interview is available here. Below, we highlight several paraphrased passages from Brenner's talk.

"The brilliant part of von Neumann's paper is in fact his description of what it takes to make a self-reproducing machine. What von Neumann shows is that you have to have a mechanism not only of copying the machine but of copying the information that specifies the machine. And this was the logical basis of self-reproduction. It highlights the fundamental error of Schrödinger, who claimed (in [5]) that the chromosomes contain the information to specify the future organism and the means to execute it. But that is not true: the chromosomes contain the information to specify the future organism and a description of the means to implement it, but not the means themselves. And that logical difference is made so crystal clear by von Neumann.

It is one of the ironies of this entire field that were you to write a history of ideas in the whole of DNA simply from the documented information, you would certainly say that Watson and Crick depended on von Neumann. Because von Neumann essentially tells you how it is done. And DNA is just one of the implementations. It is a great paradox that this connection was not seen at the time.

Biologists have yet to assimilate the whole of the theory of computation. It's an amazingly paradoxical field; most fields start struggling through from experimental confusion, through early theoretical self-delusion, finally to the great generality. And this field starts the other way around. It starts with the total abstract generality (with Gödel's hypothesis and the Turing machine) and then it takes 50 years to descend into banality."

2.1.3 Overcoming a complexity threshold

With the construction outlined above, von Neumann was able to escape the seeming paradox we mention in the introduction: if a machine $\mathcal{A}$ is to construct $\mathcal{B}$, then $\mathcal{A}$ must contain the whole description of $\mathcal{B}$ together with mechanisms that enable the construction, and thus, it must necessarily be more complex than $\mathcal{B}$.

Von Neumann explicitly discussed this seeming paradox and offered an interesting explanation of it. He suggested to associate with each machine its complexity (The complexity is deeply connected to the richness of the machine's functionality, however it remains an open problem of how to define it formally. As a very rough complexity estimate, von Neumann suggests to measure the machine's size in terms of the elementary particles). He claims that machines below a certain complexity level can only produce machines that are simpler than themselves. This agrees with our intuition leading to the paradox, as most machines we observe in the real world fall into this category. Indeed, even a 3D printer, though able to produce its own parts, is unable to make a new, completely functional 3D printer. Therefore, below a certain complexity level, the complexity of the future lineage is degenerative.

However, above a certain threshold, this limitation no longer holds. If the machine is complex enough, in particular if it can act as a universal constructor, it can construct other entities based on the instructions contained on its tape. In particular, when it receives the description of an entity more complex than itself, it is nevertheless able to build it. In this case, the paradox is resolved by off-loading the complexity onto the tape rather than into the machine, and relying on its universal capability of interpreting such instructions. Therefore, for a machine above a certain complexity threshold, the complexity of its future lineage can stay constant (in the case of exact self-replication), or can even be increasing. This is a crucial observation, alluding to a much deeper problem von Neumann aimed to solve. We will return to it in the next chapter in Section **??** and for now, we focus on a rigorous version of the self-replicator construction.

Completing the kinematic construction in full mathematical detail proved to be an arduous task, as it required addressing numerous technical difficulties related to motion and interaction in continuous space. Such challenges distracted from the real aim: understanding the logical organization of a self-replicating machine. After discussions with Stanisław Ulam at Los Alamos, von Neumann switched to a cellular model, which was far more amenable to rigorous mathematical treatment. Before we outline his rigorous solution, we define cellular automata (CAs).

2.1.4 Cellular Automata

Let $d \in \mathbb{N}$, we call $\mathbb{Z}^d$ the *d-dimensional grid* and we call its elements the cells. For a finite set S we define an *S-configuration* (or simply a configuration) of the grid to be a mapping $c : \mathbb{Z}^d \to S$. We denote the space of all S-configuration by $S^{\mathbb{Z}^d}$. Sometimes, for $v \in \mathbb{Z}^d$ we sometimes write c_v instead of $c(v)$.

A *d-dimensional neighbourhood* is a tuple $N = (n_1, n_2, \dots, n_k)$ where each $n_i \in \mathbb{Z}^d$. This induces, for each cell $v \in \mathbb{Z}^d$, a *relative neighbourhood* given by $(v + n_1, \dots, v + n_k)$. Now we can proceed with the definition of a cellular automaton.

Definition 2.3 (Cellular Automaton)**.** *A d-dimensional cellular automaton (CA) $\mathcal{A}$ is specified by a finite set of states S, a neighbourhood $N = (n_1, n_2, \dots, n_k) \subset (\mathbb{Z}^d)^k$, and a local update rule $f : S^k \to S$. These determine the CA's global update rule $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ defined, for each configuration $c \in S^{\mathbb{Z}^d}$, by*

$$F(c)(v) = f\big(c(v + n_1), \dots, c(v + n_k)\big) \quad \textit{for each cell } v \in \mathbb{Z}^d.$$

The global rule updates the state of each cell by applying the same local rule to the states in its relative neighbourhood. This update is performed synchronously for all cells in parallel. We often write $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$.

Given a CA $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ and an *initial configuration* $c \in S^{\mathbb{Z}^d}$, we can iterate the CA on c to obtain a *trajectory*

$$c \mapsto F(c) \mapsto F^2(c) \mapsto F^3(c) \mapsto \dots$$

Visualizing CA Dynamics For practical purposes, when visualizing the CA trajectories, it is often convenient to consider a *finite cyclic grid* of the form $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_d}$ for some $m = (m_1, \dots, m_d) \in \mathbb{N}^d$ and simply compute all cell indices modulo m. In this way, the dynamics take place on a discrete d-dimensional torus.

For a one-dimensional CA operating on a cyclic grid of size n with global rule F, we define the *space-time diagram of the CA with initial configuration $u \in S^n$ and t time steps* to be the matrix whose rows are u, $F(u)$, $F^2(u)$, ..., $F^t(u)$. An example of such a space-time diagram is shown in Figure 1.

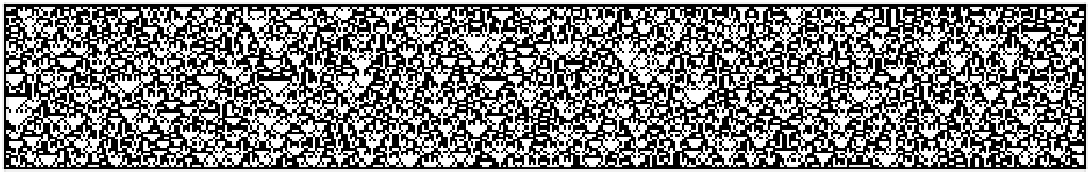

Figure 1: Space-time diagram of a 1D CA with states 0 (white) and 1 (black), radius 1, and local rule $f(x, y, z) = x + z \bmod 2$ operating on a cyclic configuration of size 250. Time flows downwards.

Two-Dimensional Cellular Automata In two dimensions, there are two neighbourhood types used most commonly: the von Neumann neighbourhood and the Moore neighbourhood, shown in Figure 2.

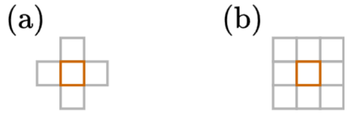

Figure 2: (a) Von Neumann neighbourhood, $N = ((0,0), (0,1), (1,0), (0,-1), (-1,0))$. (b) Moore neighbourhood, $N = ((0,0), (0,1), (1,1), (1,0), (1,-1), (0,-1), (-1,-1), (-1,0), (-1,1))$.

As the name suggests, von Neumann's neighbourhood was used in his construction of a self-replicating machine. In later chapters, we will also learn about Moore's contributions to the theory of automata.

2.2 Von Neumann's Universal Self-Replicator

Von Neumann's original construction of a self-replicator is very complex, and a fully detailed account was never completed during his lifetime. His manuscripts were subsequently edited by Arthur W. Burks and published posthumously as *Theory of Self-Reproducing Automata* [9]. The construction was later clarified and partially simplified by Thatcher [6], as presented in [2]. Thatcher preserved both the cellular automaton rule and the overall logical structure of von Neumann's design, while circumventing some highly technical issues related to the design of the tape. In this section, we present a slightly modified overview of Thatcher's approach.

2.2.1 Overview

Von Neumann defined a two-dimensional cellular automaton (with the von Neumann neighbourhood) with 29 states and demonstrated the existence of a self-replicator following the same logical principles as in his earlier kinematic model. The self-replicator is a finite pattern on the grid consisting of three main components: the supervisory unit, the tape, and the construction arm. The supervisory unit operates on the tape, implemented as a potentially unbounded line of cells in the grid, in a manner analogous to a universal Turing machine. In addition, it directs the construction arm, which can move through space and modify the states of cells around it. In this way, the system is able to construct new patterns according to the information encoded on the tape. When the pattern is supplied with its own description on its tape, it produces a copy of itself, translated to a new location in the grid. We present a schematic overview of the Von Neumann Self-Replicator (VNSR) in Figure 3.

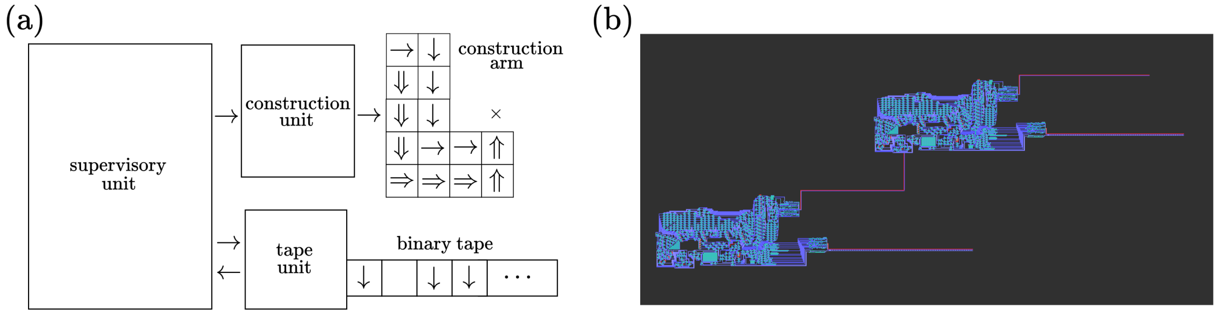

Figure 3: (a) A simplified scheme of the VNSR organisation. (b) Buckley's implementation of a simplified VNSR in the software Golly [1].

2.2.2 Von Neumann Cellular Automaton

We now present the full set of 29 states of the von Neumann cellular automaton, together with a description of its local update rule. We then show how these elementary cell states can be combined into higher-level building blocks with specific functional roles.

The first state is the quiescent state that represents "blank space" and does not influence its neighbours.

Transmission States There are 16 states dedicated to signal transmission: four ordinary transmission states (OT) represented by simple arrows directed up, down, left and right, and four special transmission states (ST) represented by double arrows in the same directions. Each of these eight directional states can be either excited or unexcited; see Figure 4 (a). An unexcited OT remains unexcited unless at least one excited OT points toward it, in which case it becomes excited in the next time step. An excited OT returns to the unexcited state unless another excited OT points toward it. Analogous rules apply to the special transmission states. These rules imply that both OTs and STs function as directed wires that transmit signals disjunctively with a one-time-step delay. Two examples of signal transmission are illustrated in Figure 4(b). We will typically represent the presence of an excited state (also called a signal pulse) by a 1, and its absence by a 0; allowing us to talk about the transmission of binary sequences.

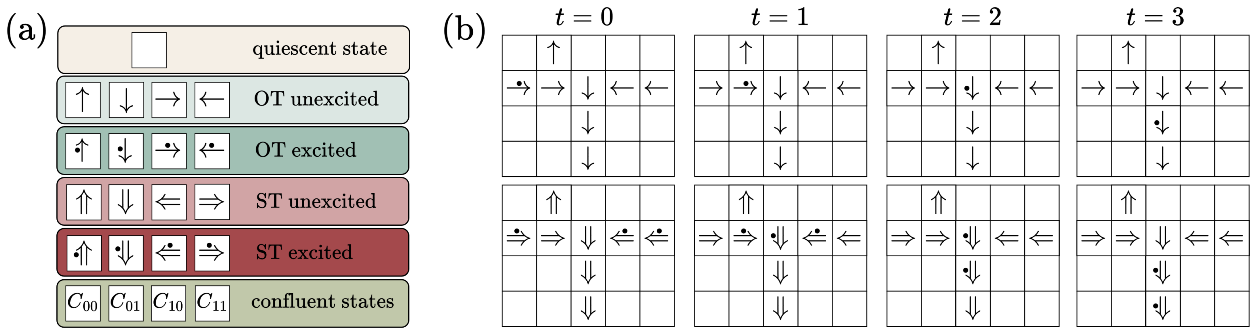

Figure 4: (a) Overview of all fundamental VNSR states: the quiescent state, ordinary transmission (OT) and special transmission (ST) states, and confluent states. (b) Top: example of signal transmission via OTs. Bottom: example of signal transmission via STs; in both examples, the central cell in the second row acts as a disjunction of incoming signals.

The OT and ST states influence each other in a destructive way: if an excited OT points towards an ST, the ST will change into a quiescent state in the next time-step, and vice versa. This mechanism is crucial for the operation of the construction arm, which must move through space without leaving residual structures behind. In the CA grid, motion is realized by creating new cell states in the direction of movement while simultaneously erasing cell states in the opposite direction.

Confluent States The second fundamental family of states consists of the confluent states, which implement several key functions: they delay signals by two time steps, act as conjunctions of incoming signals, and serve as relays for outgoing signals. There are four confluent states: $\mathrm{C}_{00}, \mathrm{C}_{10}, \mathrm{C}_{11}, \mathrm{C}_{01}$. The state C_{00} is the default, unexcited state and remains unchanged unless it simultaneously receives signals from all incoming OT states (i.e., OT states pointing directly towards it). In that case, it transitions to C_{10}. The two indices can be interpreted as memory bits of recent input activity: for a confluent state C_{xy} at time t, we have $x = 1$ if and only if all incoming OT states were excited at time $t - 1$, and $y = 1$ if and only if all incoming OT states were excited at time $t - 2$. The states C_{01} and C_{11} emit an excited signal at the next time step to all neighbouring transmission states (both OT and ST) that are not pointing toward the confluent cell. These interactions are illustrated in Figure 5.

A confluent state that receives a signal from an ST is destroyed in the next time step, becoming quiescent.

Sensitized States Finally, we introduce eight sensitized states $s, s_0, s_1, s_{00}, s_{01}, s_{10}, s_{11}, s_{000}$, which enable the creation of non-quiescent states from quiescent ones. Whenever a quiescent cell receives

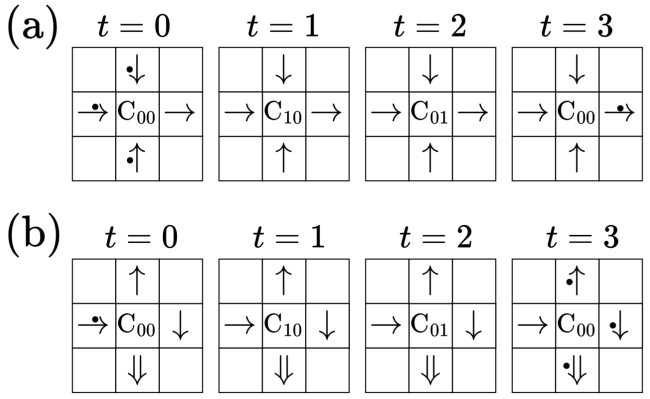
Figure 5: (a) Confluent state used as a conjunction of input OT signals. (b) Confluent state used to broadcast signals into outgoing OT and ST states.

a signal from an OT or ST state pointing towards it (possibly multiple at the same time), it transitions into the sensitized state s. Depending on the subsequent sequence of signals it receives, this state evolves through a series of intermediate sensitized states. These transitions either culminate in the creation of a specific non-quiescent state or, if the received sequence does not correspond to a valid encoding, the sensitized state eventually returns to the quiescent state. The precise transition rules are shown in Figure 6.

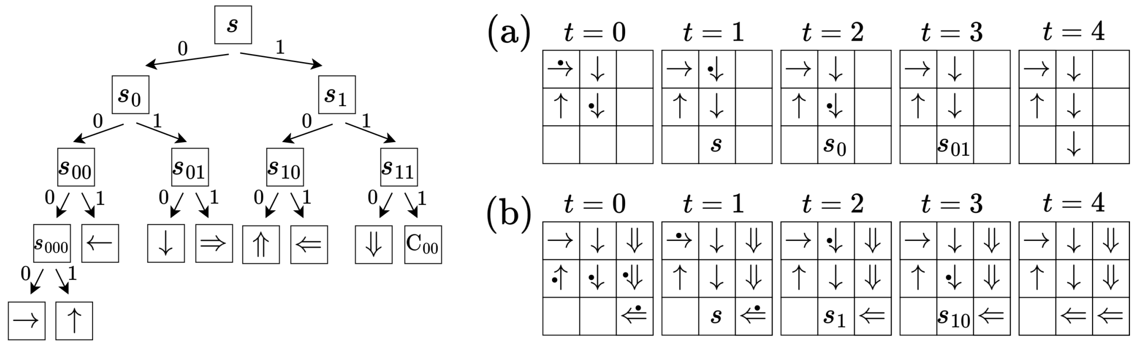
Figure 6: Left: Diagram illustrating the creation of states from a quiescent cell. Each arrow not shown leads back to the quiescent state. Right: Examples showing the creation of new states.

Remark 2.4. *A careful reader may notice that our description of the von Neumann CA does not specify the complete local update rule. Indeed, to do that we would have to list the outcomes of* 29^5 *neighbourhood state combinations. For example, we have not defined the outcome when two OTs pointing toward each other are both excited. The key point is that such interactions never occur in the operation of the self-replicator or in its subsequent copies. Consequently, many parts of the rule table can be specified arbitrarily without affecting the self-replicator's existence.*

Strictly speaking, we have therefore defined a whole class of cellular automata sharing the same behaviour on the subset of configurations relevant to the self-replicator, each of which supports a universal self-replicating machine. The full rule table would become important, however, in studying questions such as the spontaneous emergence of self-replicators from random initial configurations.

We are now ready to start constructing increasingly sophisticated functional components within the von Neumann cellular automaton.

2.2.3 Examples of Functional Units

We now present constructions of several higher-level components with specific functionality, including signal-sequence generation, sequence recognition, logical negation, memory storage, and signal wire crossings. These will form the basic building blocks of the VNSR's organization.

Pulser Let $c \in \{0,1\}^+$. The pulser P(c) is a pattern with the following behaviour: when it receives a single input signal pulse at time t, it produces the signal sequence c as output, beginning at time $t + \Delta t$, for some fixed delay $\Delta t > 0$. In Figure 7(a), we show the pulser P(1101).

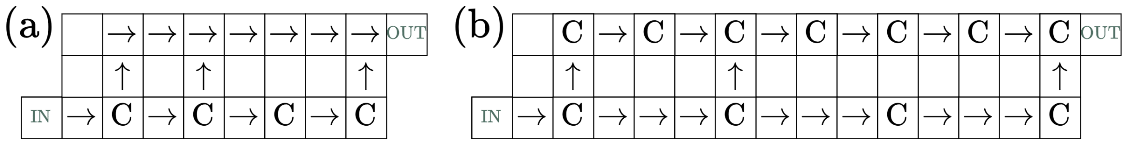

Figure 7: (a) Pulser P(1101). (b) Decoder D(1101).

Exercise 2.5 (General Pulser)**.** *Given the example above, describe the construction of* P(c) *for a general sequence* $\{0,1\}^+$*.*

Decoder Let $c, c' \in \{0,1\}^n$ be binary sequences of length $n \in \mathbb{N}$. In this section, we write $c \leq c'$ if for every $0 \leq i \leq n-1$, the implication $c_i = 1 \Rightarrow c'_i = 1$ holds.

For a fixed sequence $c \in \{0,1\}^n$ with $c_0 = 1$, the decoder D(c) is a pattern with the following behaviour: if it receives an input signal sequence c' of length n starting at time t, it produces a single output pulse (representing 1) at time $t + \Delta t$ if and only if $c' \geq c$. In Figure 7(b), we show the decoder D(1101).

Exercise 2.6 (Decoder Output)**.** *What is the output of* D(1010) *for the input sequence* 1110*?*

Exercise 2.7 (General Decoder)**.** *Using the example above, describe a construction of* D(c) *for a general sequence* $c \in \{0,1\}^+$ *containing at least one* 1*.*

The decoder serves as an intermediate step toward constructing a recognizer: a pattern that outputs 1 if and only if the input signal sequence exactly matches a given target sequence. To achieve this, we first need a mechanism that can "kill" an active signal.

Periodic Pulser A periodic pulser PP(1) is shown in Figure 8. It contains an ON (+) switch and an OFF (−) switch. When the ON switch is activated by a single pulse, the lower pulser P(11111) is triggered, and the periodic pulser begins producing a continuous stream of output pulses representing 1. When the OFF switch is activated, again by a single pulse, this active stream is "killed", and the periodic pulser returns to its original inactive state.

Signal termination is achieved using the upper pulser P(11111) in combination with the special transmission state $\Downarrow$ directed toward the active confluent state. When triggered, the first pulse travelling through the ST destroys the confluent state. The subsequent four pulses then reconstruct it in its passive state C_{00}.

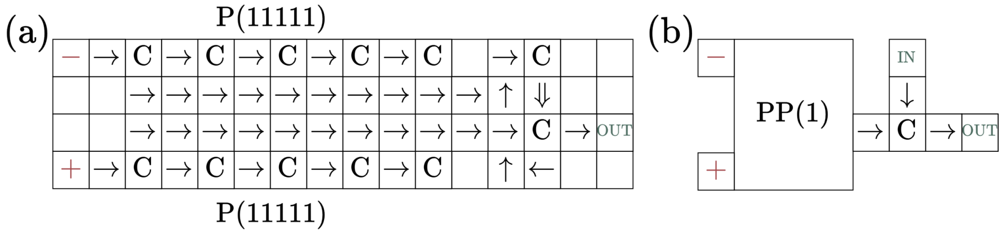

Figure 8: (a) Periodic pulser PP(1). (b) The periodic pulser can be used to store one bit of memory; the information about whether it is OFF or ON can be probed via an input signal.

Recognizer A recognizer performs a task dual to that of a pulser. Formally, let $c \in \{0,1\}^+$ with $c_0 = 1$. The recognizer R(c) is a pattern which, upon receiving an input sequence $c' \in \{0,1\}^+$ starting at time t, produces a single output pulse (representing 1) at time $t + \Delta t$ if and only if $c' = c$.

The recognizer's design combines a decoder that ensures the incoming signal has 1s in all the required positions, with a pulser that detects unwanted 1 signals in positions where c contains a 0. If such an extra 1 is detected, the pulser activates a mechanism that "kills" the output signal. As in the switch construction, signal destruction is implemented using a special transmission (ST) state. In Figure 9, we show an example of the recognizer R(1101).

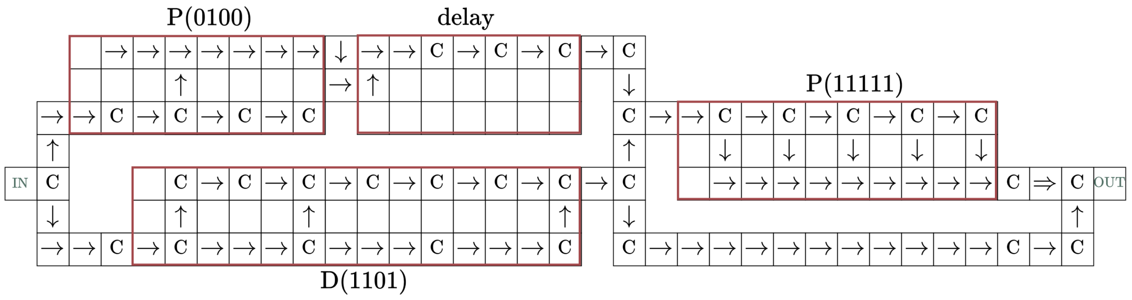

Figure 9: The recognizer R(1101).

Exercise 2.8 (General Recognizer)**.** *Using the example above, describe the construction of* R(c) *for a general sequence* $c \in \{0,1\}^+$ *containing at least one* 1*.*

Wire Crossing With pulsers and recognizers available, we can now construct more complex mechanisms, such as "wire crossings": patterns that allow signals travelling in different directions to cross without interfering with one another. An example of such a wire crossing is shown in Figure 10. We note that the crossing functions correctly only if the signals are emitted with an appropriate relative delay, ensuring that they do not enter the crossing at the same time.

Boolean Circuits Using the components described above, we are now ready to construct arbitrary Boolean circuits, represented as patterns with multiple input channels and a single output channel. To distinguish between the absence of input and the input bit 0, we encode each bit using two time steps: 0 is encoded as 10, and 1 as 11. In this encoding, the leading 1 serves as a synchronization signal indicating that a bit is being transmitted.

Exercise 2.9. *Design (at least schematically) a circuit with two input channels implementing the Boolean expression* $a \implies b$*, and a circuit implementing the expression* $(a \wedge \neg b) \vee a$*.*

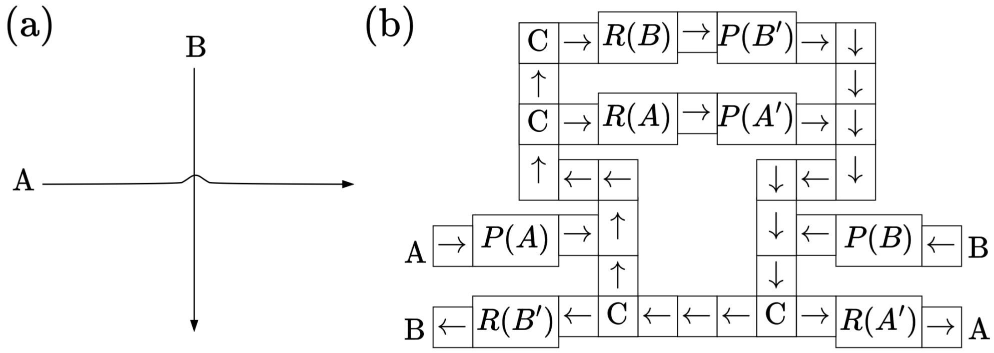

Figure 10: Wire crossing of two signals, schematically labelled A and B.

We now proceed to describe the three main components of the VNSR: the construction arm (controlled by the construction unit), the tape (controlled by the tape unit), and the supervisory unit.

2.2.4 Construction Arm

The function of the construction arm is to move within the upper-right quadrant of the plane, as shown in Figure 11(a), and to create new states from quiescent ones. It consists of two parallel rows: one composed of ordinary transmission states and the other of special transmission states. At its tip, the arm terminates in the *construction head*, a structure of three cells highlighted in Figure 11(b). The head receives signals from the construction unit through two channels, denoted O (ordinary) and S (special), as shown in Figure 11(a). These signals control both the motion of the arm and the creation of new states.

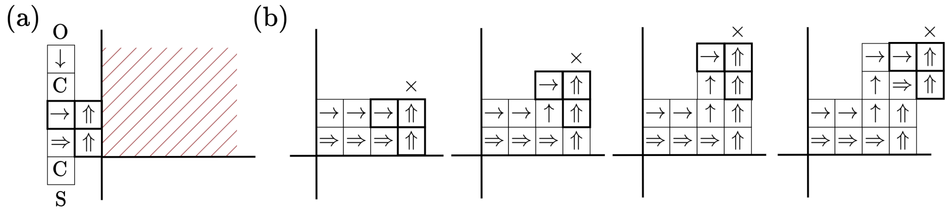

Figure 11: (a) Initial configuration of the construction arm. The red quadrant indicates the region reachable by the arm. (b) Various positions of the construction arm, with the construction head highlighted in bold. The symbol × marks the location where a new state can be created.

The construction arm has four possible movements:

1. advance upward
2. advance rightward,
3. if possible, retract downward
4. if possible, retract leftward.

Examples of these movements are shown in Figure 12. The arm can be in one of two orientations. In the *horizontal* orientation, the states adjacent to the head point toward it from the left, as in the first and last examples of Figure 11(b). In this case, the arm can retract to the left but cannot retract downward. In the *vertical* orientation, shown in the second and third examples of Figure 11(b), the arm can retract downward but not to the left.

Exercise 2.10 (Construction Arm Orders)**.** *Work out a sequence of steps, similar to those in Figure 12, that results in the construction arm retracting one step to the left.*

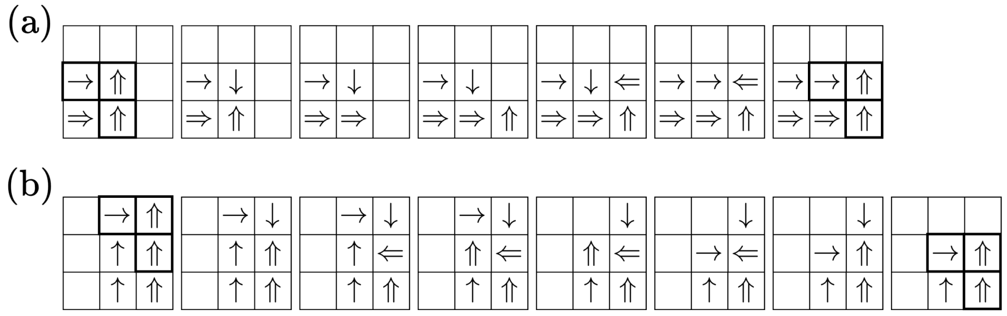
Figure 12: (a) The construction arm advances one step to the right. (b) In the vertical orientation, the construction arm can retract one step downward. Time progresses to the right; only selected time steps are shown.

The creation of a new state is achieved simply by sending the signal sequence encoding that state through the special channel. Within the red quadrant, the construction arm can build any pattern composed of unexcited OT, ST, and confluent states. It does so by first moving to the top-right cell of the target pattern and then retracting leftward, row by row, leaving the newly constructed states behind. Once the passive pattern has been assembled, it can be "brought to life" by injecting a single pulse through the special channel of the construction arm.

The construction arm cannot build arbitrary patterns: some configurations are non-constructible; such as a single excited OT in the middle of a 5 by 5 quiescent square pattern (indeed, some configurations are Garden-of-Eden, as we will discuss in a later chapter). However, it can construct any pattern consisting of passive states that can subsequently be activated by a single signal pulse. This class of patterns is already sufficiently rich and, crucially, includes the VNSR itself, as we will see later.

The Construction Unit The construction unit's function is straightforward: it receives instructions from the control unit such as "advance arm one step to the right" or "construct $\rightarrow$". Each such instruction, encoded as a bit string, is recognized, and via pulsers, transformed into a corresponding sequence of pulses for the construction arm's O and S channel.

One important detail to consider is the timing: each construction head's action can be split into an alternating sequence of pulses for the special and ordinary channel. Crucially, if a single pulse is injected both via O and S, they will reach the construction head at the same time. And this holds for an arbitrary position of the construction arm (we encourage the reader to check the cases from Figure 11 (b)). This synchronization makes it possible to alternate control between the two channels without re-timing either one. Each channel runs on a fixed, rigid clock, but effective control can switch back and forth by choosing whether a pulse appears on the ordinary channel or the special channel at a given time step.

Exercise 2.11 (Construction Arm Orders)**.** *Define a sequence of pulses that should be sent via the* O *and* S *channels, so that the construction arm creates an unexcited confluent state and then retracts one step to the left.*

2.2.5 The Tape

We implement a one-way infinite tape with binary symbols: the state $\downarrow$ represents 1, while the quiescent state represents 0. We assume that each input ends with a special delimiter sequence marking the end of the non-empty portion of the tape, allowing the Turing machine to detect the tape's boundary. The organization of the tape, including the read and write mechanisms, is illustrated in Figure 13.

The tape unit receives instructions from the control unit, such as "advance the reading head one step to the right", "retract the reading head one step to the left", "read", "write 0", and "write 1". It translates these instructions into signals transmitted to two construction arms associated with the tape. When a tape symbol is read, a return channel transmits the value of the symbol back to the tape unit.

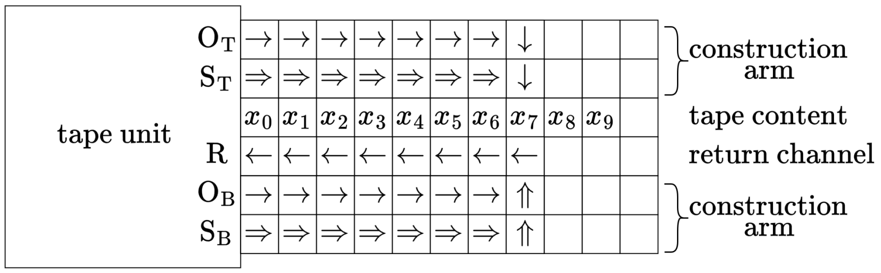

Figure 13: The organization of the tape: the top construction arm with channels O_T and S_T, the tape contents, the return channel R, and the bottom construction arm with channels O_B and S_B.

Movement of the Head: Both advancing and retracting the reading head is implemented analogously to the movement of the construction arm. The only difference is that, in addition to moving, the bottom construction arm must also update the return channel accordingly.

Reading: To read the current tape symbol x_n, the sequence 10101 is sent through the top ordinary channel O_T. If the current symbol is $\downarrow$, the sequence 10101 propagates back unchanged through the return channel. Otherwise, the subsequence 1010 transforms x_n into $\downarrow$, and the return channel instead transmits 00001. Thus, the operation destroys the information stored in x_n. This can be repaired easily: if the tape unit receives the signal 00001, it sends instructions via the top construction arm to restore x_n to the quiescent state.

Writing: To write a new symbol at position n, the instructions must work correctly regardless of the current symbol x_n. One approach would be to read x_n first and branch based on the result. However, this would introduce an unnecessary delay proportional to n (this is in contrast to the reading operation, where the lag is indeed necessary, since the supervisory unit has to wait for the outcome to determine the next action). Instead, Thatcher proposes an elegant solution: a fixed sequence of instructions that transforms x_n into the quiescent state, independent of its initial value. We leave verification of this procedure as Exercise 2.12. Once x_n has been reset, writing the new symbol and restoring the return channel are both straightforward.

Exercise 2.12 (Destroying a Tape Symbol)**.** *We first send instructions through the bottom construction arm that change the rightmost return channel state $\leftarrow$ into $\Uparrow$, resulting in the configuration shown in Figure 14(b). Next, we send the sequence* 1100001 *through the* S_B *channel at time t and through the* O_T *channel at time $t+1$ (so that both signals reach the tape symbol simultaneously). Verify that the resulting configuration is always the one shown in Figure 14(c).*

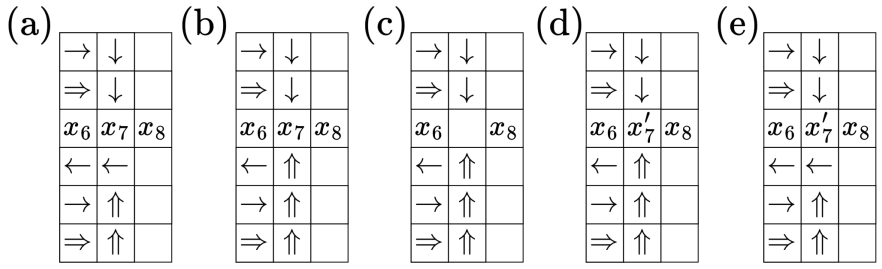

Figure 14: The process of writing a new tape symbol x'_7.

2.2.6 The Tape Unit

The function of the tape unit is conceptually straightforward. It receives instructions from the supervisory unit, such as "write 0", "read", or "move head right", encoded as binary sequences. These instructions are detected by appropriate recognizers and translated into concrete pulse sequences transmitted through the four channels $\mathrm{O_T}, \mathrm{S_T}, \mathrm{O_B}$, and $\mathrm{S_B}$.

If the instruction is to read the current tape symbol, the signal outputted by the return channel is further processed by the tape unit. Using the recognizers R(10101) and R(00001), the tape unit determines whether the tape symbol represented a 1 or a 0. This information is then re-encoded, using pulsers, into a suitable signal format and transmitted back to the supervisory unit. If the symbol read was 0, the tape unit additionally sends instructions through the $\mathrm{O_T}$ channel to restore the tape cell to the quiescent state, thereby completing the read-and-repair cycle.

2.2.7 The Supervisory Unit

The supervisory unit orchestrates the operation of the VNSR. It interfaces with a communication channel through which it sends instructions to the construction unit and the tape unit, and through which it also receives information from the tape unit. The overall organization is shown in Figure 15.

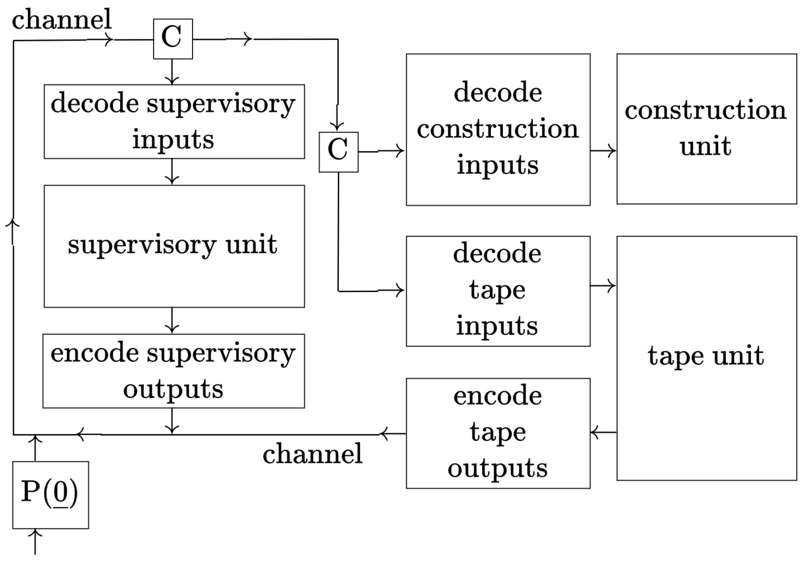

Figure 15: The VNSR components connected via the communication channel, which transmits uniquely encoded instructions.

The channel carries the following types of messages:

- **supervisory unit $\rightarrow$ construction unit**: advance arm up, advance arm right, retract down, retract left, build state.
- **supervisory unit $\rightarrow$ tape unit**: move head left, move head right, read, write 0, write 1.
- **tape unit $\rightarrow$ supervisory unit**: current tape symbol is 0, current tape symbol is 1.
- **supervisory unit $\rightarrow$ supervisory unit**: activate state i.

Each message is represented by a unique binary sequence that starts and ends with 1; such sequences are emitted by pulsers, while the corresponding decoding components are recognizers. Note that in Figure 15 the channel does not form a closed loop, preventing signals from circulating indefinitely.

Internally, the supervisory unit consists of a finite number of *state units*, with at most one state active at any given time. These state units are connected to the communication channel, as illustrated in Figure 16.

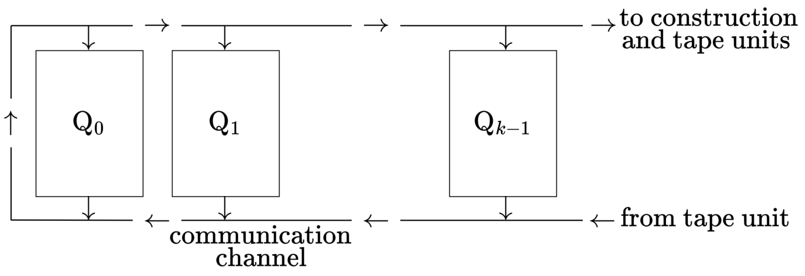

Figure 16: The supervisory unit with k state units, connected to the communication channel.

Each state unit has one of the following two roles:

- **read tape:** once activated, it sends an instruction to the tape unit to read the next symbol, and remains active until it receives a response. Depending on the outcome, it sends an instruction to activate the next state and then immediately deactivates itself; see Figure 17(a).
- **send instruction:** once activated, it emits an instruction to the construction unit or the tape unit (other than reading). It then sends an instruction to activate the next state and immediately deactivates itself; see Figure 17(b).

The sequences $\underline{\mathrm{T0}}, \underline{\mathrm{T1}} \in \{0,1\}^+$ represent the encoded tape symbols, whereas for $0 \leq i \leq k-1$, $\underline{i} \in \{0,1\}^+$ represents the encoded supervisory unit's state i.

Initially, the supervisory unit is inactive; a single pulse activates the initial state Q_0. A state unit Q_i is activated when it detects the binary sequence $\underline{i}$ via the recognizer $\mathrm{R}(\underline{i})$.

A *send instruction* state unit is straightforward: upon activation, it emits a sequence $\underline{f}_i \in \{0,1\}^+$ encoding the desired instruction (e.g., moving the construction head one cell upward). After a suitable delay, it emits a sequence $\underline{j} \in \{0,1\}^+$ encoding the instruction to activate state j, after which it becomes inactive.

The architecture of a *read tape* state unit differs. A pulse from the recognizer $\mathrm{R}(\underline{i})$ triggers both a periodic pulser PP(1) (which turns on to keep the state active) and a pulser that emits

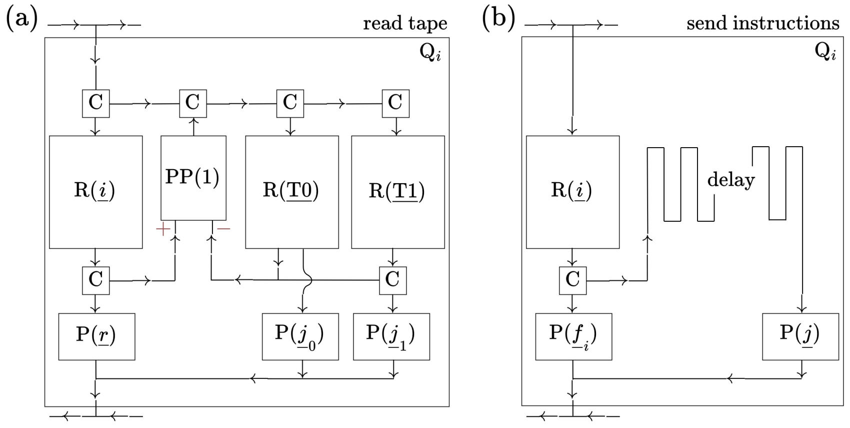

Figure 17: Schematic design of (a) a **read tape** state and (b) a **send instruction** state.

the instruction $\underline{r} \in \{0,1\}^+$ commanding the tape unit to read the current tape symbol. The periodic pulser ensures that the state unit remains active and receives the tape unit's response. Once the response arrives, and is detected by either R($\underline{T0}$) or R($\underline{T1}$), the periodic pulser is turned off. Moreover, an instruction activating the appropriate next state is emitted, and the state unit returns to inactivity.

This concludes the construction of the VNSR. For any missing details, we refer the reader to Thatcher's essay in [2].

2.3 Von Neumann's Broader Vision

It is important to emphasize that von Neumann's ambitions extended far beyond the specific construction of a self-replicating automaton in a cellular medium. To appreciate the full scope of his thinking, we strongly encourage the reader to read the five lectures he delivered at the University of Illinois in 1949, later transcribed and edited by Arthur Burks in Theory of Self-Reproducing Automata [9]. These lectures are extraordinarily rich, weaving together ideas from neuroscience, mathematical logic, computation, information theory, and thermodynamics.

In these lectures, von Neumann sought to understand both the functional organization of the nervous system and the principles governing large, complex computing systems, as well as the deep analogies between them. He envisioned the development of a new "theory of complex automata"; a program that would start by the mathematical logical treatment of the organization of perfectly functioning, discrete automata; and subsequently, shift towards an analysis of more realistic continuous and probabilistic models of computation and self-replication.

2.3.1 A Hierarchy of Self-Replicating Models

Characteristically for von Neumann, these theoretical investigations were closely tied to practical considerations. He was interested in a sequence of self-replicating constructions embedded in progressively more physically plausible media, ranging from excitation–fatigue models inspired by neurophysiology to reaction–diffusion systems closer to chemistry and biology.

Equally important was his insistence that any realistic theory must address the presence of errors. The automata he envisioned — particularly universal constructors and self-reproducing systems — are extraordinarily complex, and it is therefore essential to understand how such systems can function reliably despite error-prone components. We briefly outline these models below.

The Probabilistic Model Von Neumann emphasized that any realistic theory of self-replication must account for noise and imperfections. A viable self-replicating machine must therefore operate robustly, tolerating occasional malfunctions of its basic components so that its continued functioning, and hence survival, has non-zero probability in a noisy environment. The need for robustness becomes especially apparent when taking into account the complexity threshold von Neumann discusses: the larger the self-replicator, the more probable one if its parts will be erroneous.

In his seminal work on reliable computation [8], von Neumann identified a small collection of primitive logical elements from which any finite-state automaton can be constructed. He then analyzed a probabilistic setting in which each such component produces an incorrect output with some small probability ε; for example, a bit flip in a digital signal or a misfire in a neuron. Within this framework, he proposed architectural principles that allow reliable computation to be achieved despite unreliable components.

The Excitation-Threshold-Fatigue Model Von Neumann considered an elaboration of the cellular model in which the discrete local update rule is replaced by interactions between idealized neurons, in the spirit of the model introduced by McCulloch and Pitts [3]. In this setting, each unit represents a simplified neuron connected to its neighbors, and gets excited once the number of its active neighbours overcomes a given threshold. After firing, however, it does not immediately return to its resting state. Instead, it enters an absolute refractory period, during which it cannot be excited regardless of the activity of its neighbors. This is followed by a relative refractory period, during which excitation is again possible, but only if the number of active neighbors exceeds a second, higher threshold. This excitation–fatigue mechanism introduces temporal dynamics and history dependence into the system, bringing the model closer to biological neural behavior while still retaining a mathematically tractable structure.

The Continuous Model Von Neumann appears to have envisioned a further step toward physical realism by replacing discrete elements altogether with a fully continuous description. In such a model, the system would be governed by nonlinear partial differential equations, resembling the dynamics of diffusion and reaction processes in fluids or chemical media. The idea is that local interactions would no longer be expressed in terms of discrete states and thresholds, but through continuous variables whose evolution is determined by spatially distributed, nonlinear dynamics. This perspective anticipates later work on reaction–diffusion systems and other continuous models of pattern formation, and reflects von Neumann's broader aim of connecting logical self-replication with the laws of physics.

Conservation of Basic Quantities When working on the kinematic model, Von Neumann did intend to consider the problem of energy and force in later versions of his construction. For instance, he mentioned an intention to add "battery particles" in the environment. It becomes an interesting question whether such considerations are meaningful for the cellular model, as there are two natural points of view on such a construction: On the one hand, a non-quiescent cell can be interpreted as representing the presence of a particle. From this viewpoint, it is meaningful to study cellular automata that conserve quantities such as mass or obey other conservation laws reminiscent of physical systems. On the other hand, the cellular construction may be viewed primarily as an informational description of self-replication. In this interpretation, the automaton specifies the logical organization of the process, while an underlying physical substrate — for example, a particle system governed by conservation laws — provides the material realization. The informational dynamics then operate "on top of" this physical layer.

Arguably, the most fascinating vision of von Neumann was to use self-replicators that would not only produce exact copies, but, through inheritable mutations, create offsprings of increasing complication, resulting in a lineage with potentially unbounded complexity. This aim, at the complete core or artificial life, deserves to be properly highlighted, and we dedicate to it a substantial part of the next chapter. For now, still focusing on exact self-replication, we discuss a series of important points that might arise in a careful reader's mind.

2.4 Informal Discussion

A reader familiar with the material of the first chapter may naturally ask how self-referencing programs relate to self-replicating systems in environments such as cellular automata. In particular, the following questions arise:

- Any genuine self-replicator must deal with the problem of self-reference. This was especially apparent in the universal quine construction of the previous chapter. However, in von Neumann's design with a distinguished "constructor mode" and a "copy mode" of the self-replicator, the self-reference seems to have disappeared. Where is it hidden?
- Why is it necessary to decouple the self-replicator into a machine and its description carried on a tape? Given that von Neumann already assumes the existence of a "copy mode", could one instead design a cellular automaton pattern that, once replication begins, simply "reads itself" cell by cell and transmits this information directly to the construction arm?
- A quine may be regarded as a self-replicator, in the sense that it instructs a universal Turing machine to produce a copy of itself. Is there a fundamental difference between self-replicating programs and von Neumann's CA self-replicator? If so, in what precise sense is von Neumann's construction stronger than a quine construction? (We have yet to add a detailed discussion of this question.)

All the questions above raise important conceptual issues that are rarely discussed explicitly in the literature and are not always amenable to fully rigorous mathematical treatment. In what follows, we dedicate to each of them a detailed discussion, that will hopefully help the reader to find a resolution to these problems.

Hidden Self-Reference of von Neumann's construction

Suppose we are given a CA pattern $\mathcal{R}$, known to be capable of both universal computation and universal construction: depending on the instructions on its tape, it can perform arbitrary computation or construct a range of CA patterns. Suppose, however, we were not given any information about the internal functioning of $\mathcal{R}$. This is precisely the setting in which Kleene's recursion theorems would allow us to establish that $\mathcal{R}$ can self-replicate. Indeed, using Kleene's theorems, one could encode on the tape the following instructions:

1. construct the pattern with description $[\mathcal{R}]$;
2. copy the entire content of this tape onto the tape of the newly constructed pattern.

This yields a computationally universal recipe for self-replication: independently of the concrete design of $\mathcal{R}$, there exists an enriched quine, guaranteed by Kleene's recursion theorems, that turns the constructor into a self-replicator.

This, however, is not von Neumann's strategy. Rather than treating $\mathcal{R}$ as an arbitrary universal constructor, von Neumann explicitly designs it from elementary components and separates $\mathcal{R}$ into two distinct subsystems: a universal constructor $\mathcal{B}$, which interprets the tape and builds the pattern it describes, followed by a copying unit $\mathcal{A}$, which copies the tape. As a result, part of the burden of self-reference is shifted from the tape to the machine's architecture itself.

From the perspective of quine constructions, this design choice has an important consequence: once $\mathcal{R}$ enters its copy mode, the specific content of the tape becomes largely irrelevant, since whatever is written on it will be duplicated. In this sense, the tape does not function as a quine by virtue of its content alone. An analogy would be to claim that a page containing arbitrary text is a quine simply because, when placed in a copy machine, it is reproduced.

In von Neumann's construction, self-reference only becomes apparent when the machine $\mathcal{R}$ and a tape containing its own description are considered as a single composite system. The tape must encode the full description of $\mathcal{R}$, including, in particular, the copying unit $\mathcal{A}$ and its activation at the appropriate stage of the construction process. It is at this level—the interaction between machine and description—that self-reference reappears.

To summarize, once the design of von Neumann's machine is fixed, the quine that appears on the tape does not come from a universal recipe. In fact, it is heavily fine-tuned to the machine's design, in particular, it relies on the copy mode of the machine. However, the construction as a whole: i.e., the machine together with its tape is universal with respect to the laws of physics governing the system's dynamics. Indeed, von Neumann did not choose the CA design to contain any particular mechanism to be leveraged by his constructions. He simply wanted to identify rules that are expressive enough to allow for a higher-level construction of embodied Turing machines. As such, we suggest the following separation:

- quines can come from universal recipes with respect to universal Turing machines that execute them
- embodied self-replicators can come from universal recipes with respect to the underlying laws of physics

Without a doubt, von Neumann's design represents such a universal recipe. In this sense, we can view his design as a physical analogue of the Kleene's recursion theorems.

Dimensionality Nuances: Is the tape even necessary?

Why was it necessary in von Neumann's construction to decouple the self-replicator into a machine and its description contained on a tape? Since we already assume in von Neumann's construction that a part of the machine is a copy mechanism, we could just build a machine that does whatever it needs to do, and once it starts to self-replicate, it just reads itself, cell by cell, sending the state information through the construction arm directly.

Such a construction would be feasible for a particular CA rule design; however, it would not yield a "physically universal" solution in the way von Neumann's design does. In the tape-less design, we would have to assume the machine's ability to read its own description in a non-destructive way, even if the machine has the same dimensionality as the space in which it exists. However, for example, this does not seem to be immediately available in the three-dimensional world we live in. It seems crucial to associate with an organism a description of lower dimensionality. In particular, if the description is one-dimensional, it becomes straightforward to read it sequentially. It does not seem too much to assume that the laws of physics enable reading a one-dimensional tape embedded in a higher-dimensional space. It is intriguing that this is exactly the solution biology came up with.

In principle, the description could have been two-dimensional, but then the mechanism copying it would have had to evolve in a more sophisticated way.

2.5 Chapter Summary

In this chapter, we have studied in detail von Neumann's seminal work on self-replication, which predicted the logical principles of biological self-replication. Von Neumann's construction provides a solution to our initial question of how to build machines that produce exact copies of themselves, in an abstract model known as the cellular automaton. This solution does not immediately translate to real-world self-replicators, and von Neumann intended to address this issue by extending his constructions to a progression of models of increasing physical feasibility. Crucially, von Neumann's vision extended far beyond exact self-replication: ultimately, he was interested in designing replicators that can build offspring of higher complexity than themselves, resulting in a lineage of self-replicators of growing complexity, akin to the way living systems evolve. This aim will be the central topic of the next chapter, where we also review the subsequent work on self-replicating constructions.

2.6 Further Recommended Resources

- **On the history of discoveries related to the DNA structure and von Neumann:** `https://www.youtube.com/watch?v=_w2453iom9s`.
- **Theory of self-reproducing automata [9]:** a book on von Neumann's work edited by Burks which includes von Neumann's five lectures at the University of Illinois covering, among other things, the organization of the kinematic model; as well as von Neumann's original vision for the VNSR in the cellular environment.
- **Essays on Cellular Automata [2]:** A collection of essays curated by Burks deeply connected to the theory of automata and self-replication, including Thatcher's simplified VNSR construction.
- **On the significance of von Neumann's construction for open-ended evolution [4].**
- **Clip from a documentary on John von Neumann:** `https://www.youtube.com/watch?v=Oh31I1F2vds`.

References

[1] Buckley, W. R. Signal crossing solutions in von Neumann self-replicating cellular automata. In *Automata* (2008), pp. 453–503.

[2] Burks, A. *Essays on Cellular Automata.* University of Illinois Press, Urbana, 1970.

[3] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[4] McMullin, B. John von Neumann and the Evolutionary Growth of Complexity: Looking Backward, Looking Forward. *Artificial life 6* (02 2000), 347–61.

[5] Schrödinger, E. *What Is Life? The Physical Aspect of the Living Cell.* Cambridge University Press, Cambridge, United Kingdom, 1944. Based on lectures delivered at Trinity College, Dublin.

[6] Thatcher, J. Universality in the von Neumann cellular model. *Essays on Cellular Automata* (1970).

[7] von Neumann, J. The General and Logical Theory of Automata. In *Cerebral Mechanisms in Behavior: The Hixon Symposium*, L. A. Jeffress, Ed. John Wiley & Sons, New York, NY, USA, 1951, pp. 1–41. Read at the Hixon Symposium, Pasadena, 20 September 1948.

[8] von Neumann, J. Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. In *Automata Studies*, C. E. Shannon and J. McCarthy, Eds., vol. 34 of *Annals of Mathematics Studies.* Princeton University Press, Princeton, NJ, 1956, pp. 43–98.

[9] von Neumann, J. *Theory of self-reproducing automata.* University of Illinois press Urbana, 1966, Editor: A.W. Burks.