Self-replication and Computational Universality

Jordan Cotler* Harvard University
Clément Hongler† EPFL
Barbora Hudcová‡ EPFL

October 10, 2025

Abstract

Self-replication is central to all life, and yet how it dynamically emerges in physical, non-equilibrium systems remains poorly understood. Von Neumann's pioneering work in the 1940s and subsequent developments suggest a natural hypothesis: that any physical system capable of Turing-universal computation can support self-replicating objects. In this work, we challenge this hypothesis by clarifying what computational universality means for physical systems and constructing a cellular automaton that is Turing-universal but cannot sustain non-trivial self-replication. By analogy with biology, such dynamics manifest transcription and translation but cannot instantiate replication. More broadly, our work emphasizes that the computational complexity of translating between physical dynamics and symbolic computation is inseparable from any claim of universality (exemplified by our analysis of Rule 110) and builds mathematical foundations for identifying self-replicating behavior. Our approach enables the formulation of necessary dynamical and computational conditions for a physical system to constitute a living organism.

*Email: jcotler@fas.harvard.edu.
†Email: clement.hongler@epfl.ch.
‡Email: barbora.hudcova@epfl.ch.

Contents

1 Introduction

The emergence of self-replicating structures in Earth's oceans and their evolution into the diversity of life we witness today remains one of science's greatest mysteries. This gap in understanding prevents us from recreating the process through simulation or experiment, raising fundamental philosophical and mathematical questions about life's nature. In the 1940s, John von Neumann [1] illuminated self-replication's key principles using an abstract framework known as cellular automata (CAs) [2]. As illustrated in Fig. 1(a), physical or chemical systems can be abstracted as CAs that, under the right conditions, dynamically produce self-replicating organisms. Our ultimate goal is to identify which dynamics and configurations permit the emergence of self-replication. Von Neumann's pioneering work provides a foundation: he designed a CA with a non-trivial self-replicator leveraging the dual use of an organism's description – once to be interpreted for construction and once to be transcribed to the new offspring. Remarkably, von Neumann's model distinguished between replication and transcription before DNA's structure was discovered [3].

Von Neumann's work highlights a key distinction between genuine self-replication and superficial copying. Consider red dye diffusing in water: though color spreads throughout, we would not call dye a 'self-replicating organism.' To exclude such examples, von Neumann designed a universal self-replicator: a finite CA pattern that performs arbitrary computations on its input tape and constructs arbitrary patterns from tape descriptions. When reading its own description, the machine builds an exact copy of itself. Thus, the underlying CA rules must be 'Turing-complete', capable of arbitrary computation. Crucially, self-replication emerges from the machine's computational universality.

There have been many simplifications of von Neumann's construction [4, 5, 6, 7], including most famously Conway's Game of Life [8] which features deceptively simple rules generating viscerally lifelike structures [9]. Game of Life was believed to contain self-replicators before it was formally proven, due to its Turing completeness [10, 11]. Wolfram [12] later emphasized that Turing-complete CAs exhibit distinctive qualitative features, prompting his conjecture that Rule 110 – a one-dimensional automaton with an extremely compact description – was Turing-universal based on these characteristics, subsequently proven through elaborate analysis of the rule's dynamics [13, 14, 15].

However, a fundamental ambiguity persists in these core notions [16, 17]. Turing constructed a universal Turing machine without formally specifying the notion of Turing completeness [18]. Similarly, von Neumann constructed a non-trivial self-replicator without formally characterizing non-trivial self-replication. A natural hypothesis is that Turing universality implies self-replication [4] – after all, such systems support quines, programs outputting their own description [19, 20]. However, as we show, nuances about how physical systems perform computation can matter profoundly.

We formalize two kinds of Turing universality for CAs: local universality – an automaton's ability to simulate any Turing machine (implicitly assumed by von Neumann and Wolfram), and global universality [21] – the ability to simulate any same-dimension CA. Local universality focuses on computational power within the CA [22], while global universality concerns emulating the entirety of other physical systems [16].

We examine the relationships between local and global simulation and establish their precise connections with self-replication. Though most natural examples like Game of Life exhibit both local and global universality [23], the distinction between these two forms of universality becomes fundamental when it comes to self-replication. Our analysis addresses several key questions: whether local and global universality are equivalent (they are not, global is stronger), whether local universality implies self-replication (it does not), whether global universality implies self-replication (it

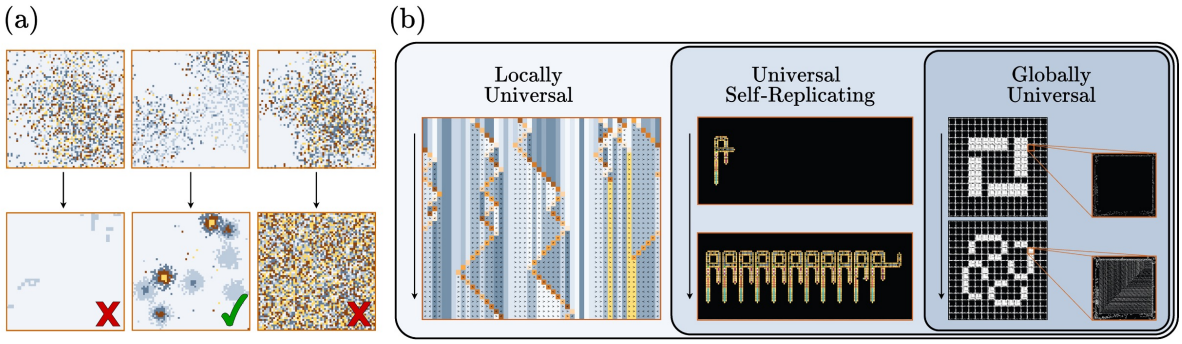


Figure 1: (a) Physical or chemical systems, abstracted as cellular automata (CAs), may in appropriate circumstances evolve from random initial conditions into self-replicating organisms. We develop a theory constraining which dynamics and configurations can give rise to self-replicating organisms. (b) We rigorously define what it means for a CA to be Turing-universal and establish its relationship to self-replication. We identify and study three classes of CAs with increasing capabilities: (i) locally universal CAs can host a universal Turing machine within their dynamics, (ii) universal self-replicating CAs support self-replicating structures that also perform Turing-universal computation, and (iii) globally universal CAs can simulate any other CA of the same dimension via local, translation-invariant encodings. We prove these form a strict hierarchy: Locally Universal $\supset$ Universal Self-Replicating $\supset$ Globally Universal. Consequently, some CAs (such as the one shown in the 'locally universal' block) are Turing-universal and yet provably cannot support self-replicating organisms, analogous to biological systems capable of transcription but not replication.

does), whether Rule 110 supports self-replication (unknown), and whether dimensionality affects these relationships, summarized in Fig. 1(b).

Our work establishes foundations for a mathematical theory of self-replication and its connections to computability and complexity theory. By disentangling self-replication's interface with computational universality, we identify mechanisms essential for replication vis-à-vis transcription. Our definitions and results enable the specification of precise, necessary conditions for physical systems to constitute self-replicating organisms.

2 Results

Section 2.1 defines local and global universality, demonstrating the latter is strictly stronger via reversible CAs [24]. We verify that our definitions align with previous CA constructions, including Rule 110's elaborate Turing completeness proof. Section 2.2 presents necessary conditions for non-trivial self-replication and modifies von Neumann's construction to demonstrate 1D universal self-replicators. Section 2.3 constructs a 1D locally universal CA that cannot sustain non-trivial self-replication, as calibrated by our necessary conditions.

2.1 Computational universality in physical dynamics

We use CAs to model spatially local, translation-invariant physical dynamics. A CA consists of a regular grid where each cell can be in one of finitely many states, evolving synchronously according to local rules that depend only on each cell's current state and its neighbors. Rule 110 [12] is a famous 1D example with black and white cells where each cell's next state depends on its current

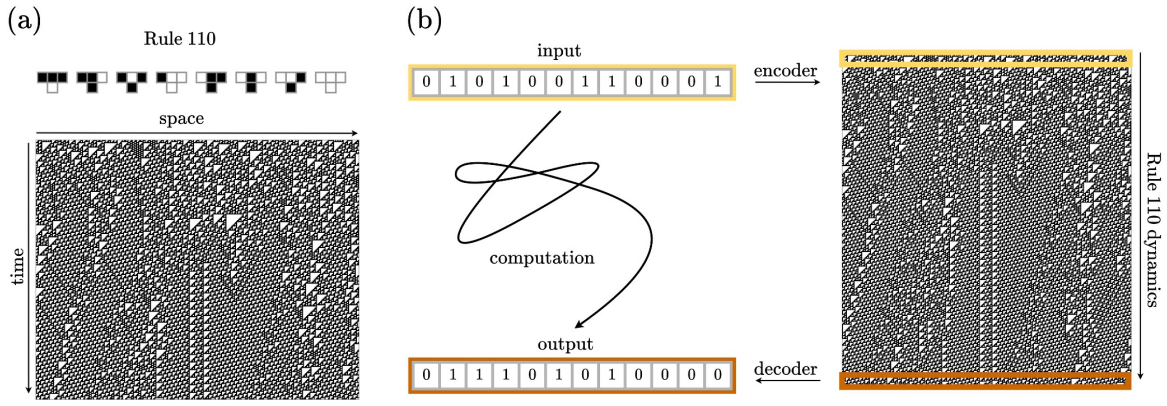


Figure 2: (a) Space-time diagram of the elementary CA, Rule 110. Each cell assumes a binary state (black or white) and evolves deterministically based on its current state and those of its two nearest neighbors. Time progresses downward from a random initial configuration (top row), revealing the emergence of complex spatiotemporal structures. (b) Schematic of a computational process, as implemented by a Turing machine, where an input bit string undergoes computation to produce an output bit string. To simulate this computation using Rule 110, three requirements must be satisfied: (i) the input bit string must be encoded into the CA's initial configuration, (ii) the CA's evolution must faithfully track the Turing machine's computational steps, and (iii) a decoding scheme must extract the final output from the CA dynamics such that it matches the Turing machine's output.

state and nearest neighbors, as shown in Fig. 2(a).

We examine the relationship between dynamics and computation: in what sense does physical dynamics perform computation? This question, originating with Turing [18], has a rich history [25]. The Church-Turing Thesis stipulates that a universal Turing machine can simulate any physical system's dynamics, including another Turing machine, but is vague about what constitutes simulation. When you run Python code, symbolic manipulation is encoded into electron configurations in semiconductors, evolving according to physical laws, and then decoded back to symbolic output. This illustrates the necessity of 'encoding' and 'decoding' operations mediating between symbolic computation and physical dynamics (Fig. 2(b)). Physical dynamics instantiate specific symbolic computation if an encoder-decoder pair exists relating them, subject to sensible constraints. Crucially, the encoder and decoder must be computationally simpler than the computation attributed to the physical system. Otherwise, we risk computational misattribution where interesting computation is performed by the encoding/decoding apparatus rather than the physical dynamics.

We define two forms of CA Turing-universality, starting with:

Definition 2.1 (Local universality of a CA, informal)**.** *A CA is* locally universal *if it can simulate a universal Turing machine in a bounded region whose size scales with the machine's tape usage. Specifically, there exists an encoder-decoder pair that maps between universal Turing machine dynamics and CA dynamics, where both the encoder and decoder have time complexity bounded by a function of the size of the non-blank portion of the Turing machine tape.*

The key idea is that a physical system is Turing-universal if it can host a localized universal computer that expands as memory requirements grow (Fig. 3(a)). Additional technical details on time slowdowns, symmetry transformations, and background configurations appear in the Appendix.

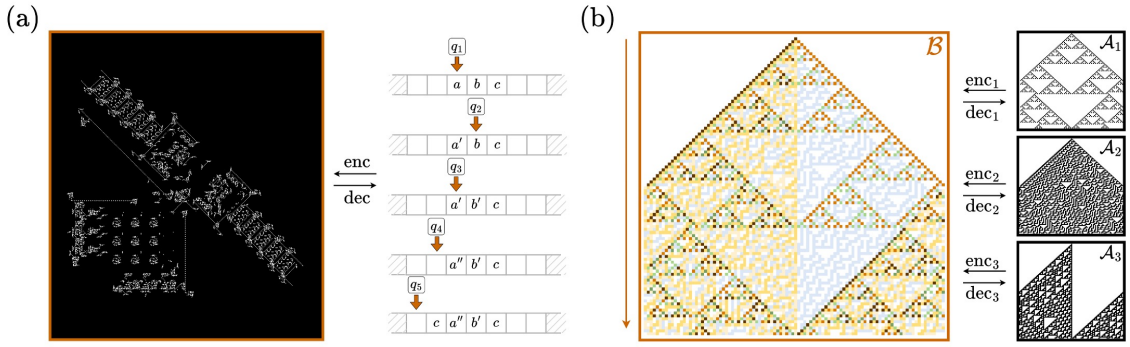


Figure 3: (a) Local universality in a 2D CA. A spatially finite configuration (shown at a fixed time) can perform Turing-universal computation: any Turing machine can be encoded into such a configuration, with the CA's evolution tracking the machine's computation and its states recoverable through decoding. The CA is termed *locally universal* because Turing machines with finite input can be simulated within finite (though potentially growing) spatial regions, rather than requiring infinite configurations of the CA at the outset. (b) Global universality in a 1D CA. The CA $\mathcal{B}$ (shown evolving top to bottom) can simulate infinite configurations of any other 1D CA through local, translation-invariant encoding and decoding maps. By contrast, local universality only requires a CA to be able to simulate a Turing machine within a finite region. Here we exemplify how $\mathcal{B}$ globally simulates dynamics in three different CAs $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ simultaneously.

Definition 2.2 (Global universality of a CA, informal)**.** *A CA is* globally universal *if it can simulate any other CA of the same or lower dimension. The encoder and decoder must be block encodings and thus have bounded time complexity for encoding/decoding between local regions.*

This captures a fundamentally different notion: a globally universal physical system can simulate any other physical system while preserving spatial locality, i.e. local interactions in the simulated system correspond to local interactions in the simulating system (see Fig. 3(b)). This is stringent because CAs are infinite in extent; we are characterizing how one infinite system can simulate the entirety of another. Global universality is a form of 'intrinsic universality' in the CA literature, formulated in several variants [21, 26]. By contrast, despite local universality's foundational importance to von Neumann's and Wolfram's work, the field has lacked a rigorous definition. Our framework fills this gap.

What, then, is the relationship between local and global universality? Let $\mathsf{LocallyUniversal}$ denote the set of locally universal CAs and $\mathsf{GloballyUniversal}$ denote the set of globally universal CAs. We have:

Theorem 2.3 (Computational universality hierarchy for CAs)**.** $\mathsf{GloballyUniversal} \subsetneq \mathsf{LocallyUniversal}$.

This reveals a hierarchy of computational power. Every globally universal CA must be locally universal: if a CA can simulate any CA, it can simulate one implementing a Turing machine locally. The proof verifies that global simulation of a locally universal CA yields a valid local encoder-decoder pair for the simulating CA (see Appendix). The containment is strict: some CAs are locally universal but not globally universal. Consider reversible CAs, which run equally well forwards or backwards. Some can locally implement reversible universal Turing machines but cannot globally simulate irreversible CAs [27], proving $\mathsf{GloballyUniversal} \subsetneq \mathsf{LocallyUniversal}$.

The literature contains numerous CAs exhibiting either local or global universality [21, 28, 29, 11, 13]. Much effort has focused on constructing CAs with simple rules that locally implement

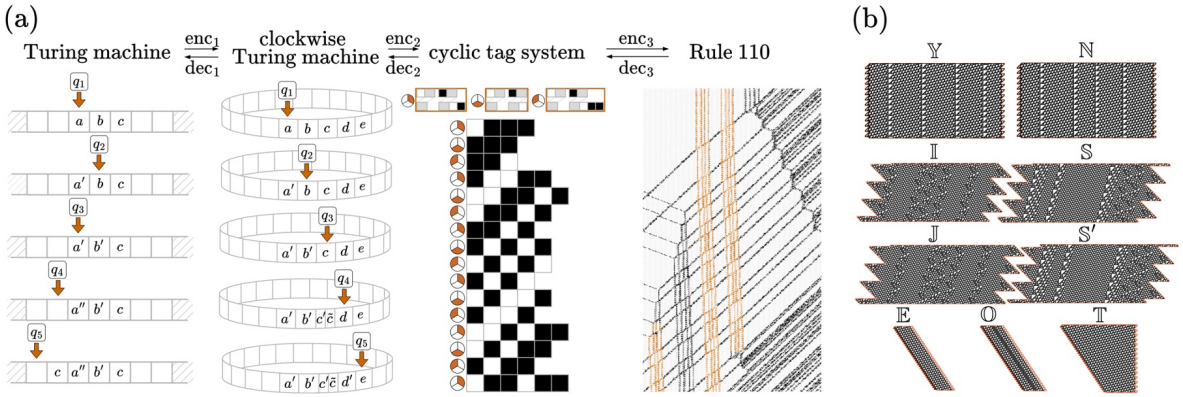


Figure 4: (a) Rule 110 can simulate any Turing machine via a sequence of intermediate encodings and decodings. In particular, Rule 110 (where certain "gliders" are colored orange for emphasis) simulates a cyclic tag system which itself simulates a clockwise Turing machine, which itself simulates an ordinary Turing machine. We carefully keep track of the computational complexities of the encodings and decodings as well as their compositionality properties, and establish that the composed encoding and decoding between Rule 110 and a Turing machine requires only polynomial time complexity and linear space complexity. (b) The encoding from a Turing machine to Rule 110 and the subsequent decoding are very intricate. Shown here are encoding components of Turing machine states mapping to modular regions of the Rule 110 CA that fit together like puzzle pieces.

universal Turing machines [13]. However, these constructions often leave encoder-decoder pairs implicit and neglect their computational complexity. This oversight can be critical: without proper constraints, complex encoders and decoders can spuriously perform the computation we want to attribute to the CA itself [23, 30]. The elementary Rule 110 provides a compelling case study. This 1D CA with nearest-neighbor interactions simulates a universal Turing machine through a remarkable construction, yet the encoder-decoder computational complexity was never analyzed. Building on previous work [13, 14, 15], we establish:

Theorem 2.4 (Local universality of Rule 110)**.** *Rule 110 is locally universal with respect to an encoder-decoder pair with polynomial time complexity and linear space complexity. Rule 110 incurs a polynomial time slowdown relative to the universal Turing machine it simulates.*

Our analysis traces a chain of simulations: universal Turing machine ↔ clockwise Turing machine ↔ cyclic tag system ↔ Rule 110 (Fig. 4(a)). By analyzing the computational complexity at each step and exploiting transformation compositionality, we bound the overall encoding/decoding complexity. This rigorous analysis necessitates making all encodings and decodings fully explicit (Fig. 4(b)), providing a complete characterization of Rule 110's computational universality.

2.2 Necessary conditions for self-replication

How does self-replication interface with computational universality? What logical organization is necessary or sufficient for a finite CA pattern to self-replicate? Von Neumann addressed sufficiency by constructing a universal self-replicator in a 29-state, 2D CA. His work and its refinements [1, 31, 32] demonstrated self-replication in CAs. Here, we focus on necessary conditions for CAs to support self-replicating configurations. A natural question from von Neumann's construction is whether two dimensions are necessary for universal self-replication. We establish:

Theorem 2.5. *Universal self-replicators exist in one dimension.*

We modify von Neumann's 2D design so that its 'constructing arm' builds replicas with a horizontal rather than diagonal offset, allowing operation within a bounded-height strip. Encoding this bounded-height strip into a 1D CA with an expanded state space (more than 29 states) yields a 1D CA implementing universal self-replication.

Von Neumann's Turing-universal constructor ensures self-replicating subsystems perform dynamics beyond mere copying. However, von Neumann did not articulate necessary conditions for CAs to support universal self-replicators; he provided an example but did not define the general set of CAs UniversalSelfReplicating. We define a necessary condition for universal self-replication from CA dynamics, phrased informally below.

Definition 2.6 (Local self-replicating condition, informal)**.** *A CA satisfies the* local self-replicating condition *if it contains a bounded pattern that evolves for at least time $T \geq 2$ and proliferates arbitrarily many copies, where each copy exhibits the same bounded dynamical patterns.*

Requiring $T \geq 2$ ensures dynamic rather than static patterns. This captures non-trivial self-replication as unbounded proliferation of dynamical patterns.

The local self-replicating condition represents a minimal requirement for non-trivial self-replication. Therefore:

Conditions 2.7 (Necessary conditions for universal self-replicators, informal)**.** *Any CA in* UniversalSelfReplicating *is locally universal and satisfies the local self-replicating condition.*

These requirements reflect that universal self-replication must be non-trivial (containing dynamical components) and computationally universal. While additional criteria are needed to fully characterize UniversalSelfReplicating, these conditions suffice for our analysis. Von Neumann's universal constructor exemplifies both conditions.

There exist simpler examples of self-replicating CA subsystems that are not themselves Turing-universal, and yet still exhibit non-trivial dynamics. Simpler examples like Langton's loops [4] through the Turing-universal Perrier-Sipper-Zahnd loop [33] all satisfy our local self-replicating condition, suggesting a hierarchy ordered by dynamical complexity. Such observations motivate the development of more stringent conditions that capture increasingly sophisticated dynamics beyond mere periodicity, potentially enabling a systematic classification scheme for self-replicating systems based on their computational and dynamical properties.

2.3 Self-replication versus computational universality

Does local universality, namely the ability to build a local Turing-universal computer within a CA, guarantee non-trivial self-replicating configurations? Surprisingly, we prove that the answer is no. We construct a locally universal CA that cannot support self-replication, demonstrating a separation between computational power and reproductive capability. Our CA achieves universality through isolated computational elements whose restricted communication precludes self-replication.

We first construct a 1D, Turing-universal 'talking heads' CA, with mobile Turing machine heads that communicate through proximity or shared tape. We then modify this to a 'non-talking heads' CA, enforcing that proximal heads halt immediately. We further prevent tape-mediated communication by appending directional markers to the state space: when a head moves rightward from a cell it marks that cell with a $\rightarrow$, and when moving leftward it marks with a $\leftarrow$. These markers establish exclusive territories since any head encountering a cell marked by another head halts immediately. This blocking of information transfer prevents self-replication:

Theorem 2.8 (Non-talking heads CA does not have self-replication)**.** *A non-talking heads CA does not satisfy the local self-replicating condition and therefore cannot belong to* UniversalSelfReplicating*.*

This shows UniversalSelfReplicating and LocallyUniversal to be distinct, as the non-talking heads CA belongs to the latter but not the former. More broadly:

Theorem 2.9 (Self-replication lies strictly between global and local universality)**.** *The following strict inclusions hold:* GloballyUniversal $\subsetneq$ UniversalSelfReplicating $\subsetneq$ LocallyUniversal*.*

This hierarchy (Fig. 1(b)) reveals fundamental relationships between computational power and self-replication. For the proof, GloballyUniversal $\subsetneq$ UniversalSelfReplicating holds since globally universal CAs can simulate any CA preserving spatial locality. The inclusion is strict because reversible CAs support universal self-replication yet cannot globally simulate irreversible dynamics (see Appendix). The inclusion UniversalSelfReplicating $\subsetneq$ LocallyUniversal follows from Condition 2.7, with strict inclusion from Theorem 2.8.

Our results clarify that local computational universality is necessary but insufficient for non-trivial self-replication. The non-talking heads construction abstracts biological concepts of transcription and replication, showing systems can possess transcriptional capabilities without achieving true replication. As alluded to earlier, our result may seem counterintuitive given the existence of quines, which are programs in any Turing-complete language outputting their own source code [20, 34]. However, a quine produces a description of the copying mechanism without replicating the mechanism itself. In our framework, this is the difference between a head copying tape regions versus creating new heads or recruiting existing heads. Quines represent information copying; genuine self-replication requires reproducing the physical copier itself, explaining why computational universality cannot guarantee self-replication.

3 Discussion

Our work clarifies the relationship between computational universality and self-replication in physical systems. A key contribution is our rigorous treatment of encoder-decoder complexity in defining local universality, generalizing the work of [30]. This is exemplified by establishing polynomial-time encoding/decoding bounds for Rule 110. The complexity constraint prevents computational misattribution, ensuring computation is genuinely performed by CA dynamics rather than sophisticated encoding/decoding schemes. We demonstrated that local Turing-universal computation does not necessarily enable self-replication, a counterintuitive finding since universal computation might seem to enable all reproductive capabilities. Our non-talking heads CA abstracts the biological principle that systems capable of transcription may lack coordinated functionality for replication. At a technical level, identifying 'head' and 'tape' structures shows that inter-head communication is crucial for replication.

Many challenges remain. Determining whether specific CAs support universal self-replication is open, and Rule 110 presents a compelling test case:

Conjecture 3.1. *Rule 110 is in* UniversalSelfReplicating*.*

Beyond specific examples, sharpening our definition of self-replicating systems is essential, as our local self-replicating condition is only a minimal requirement. Developing increasingly stringent conditions would clarify which dynamics enable sophisticated reproduction. These may depend on CA dimensionality, though our 1D universal self-replicator demonstrates that dimension does not limit self-replication capacity. Further incorporating environmental noise is essential for understanding robustness of self-replication under realistic conditions.

Our work establishes a theoretical foundation for understanding self-replication beyond isolated examples, presenting the first general results on the relationship between computational universality and self-replication. These advances represent initial steps toward a comprehensive theory that will require integrating insights from non-equilibrium physics, chemistry, and computer science. More broadly, our work lays the ground for formal explorations of abstract mechanisms that lead to the emergence of life.

Acknowledgments

JC would like to thank Bert Chan, Semon Rezchikov, and Jensen Suther for valuable discussions. JC is supported by a fellowship from the Alfred P. Sloan Foundation. BH and CH would like to thank Vassilis Papadopoulos, João Penedones, Jakub Krásenský, and Jiří Tůma. The UniversalSelfReplicating and GloballyUniversal images in Fig. 1(b) and the Game of Life Turing machine construction in Fig. 3(a) were generated using an open-source application Golly [35].

A Related work

A.1 Self-replicators in cellular automata

The foundational theory of self-replication in CAs was developed by John von Neumann, who designed a universal self-replicator in a 2D automaton. His construction, published posthumously by Burks [1] and later completed in detail by Thatcher [31], is estimated to require between 50,000 and 200,000 cells. Subsequent work, termed the 'second generation of self-replicating automata research', aimed at reducing the complexity of von Neumann's model while preserving its computational universality. Codd reduced the number of CA states from 29 to 8 [32], while Arbib simplified aspects of the construction logic at the expense of introducing a more elaborate transition function [36].

The third generation began with Langton, who argued that Turing universality is a sufficient condition for non-trivial replication, but not a necessary one. He constructed 2D loops [4] that are not computationally universal but implement self-replication via a mechanism analogous to von Neumann's construction, through the transcription of the replicator's description. As such, Langton's loops are deemed non-trivial. Unlike von Neumann's description of a large scale self-replicator, Langton's loops are small (86 cells within an 8-state CA), which makes them the first explicit implementation of non-trivial self-replication. Langton's work sparked extensive subsequent research on compact replicating structures.

One branch of this research pursued further simplification: Byl [5] designed a 12-cell loop in a CA with 6 states, Reggia et al. [37] produced a 5-cell replicator in a 6-state CA, Ibáñez et al. [38] developed loops where, interestingly, replication is based on self-inspection rather than having the replicator's description explicitly available as was the case in previous constructions. Morita and Imai [39] designed self-replicating loops in a reversible cellular automaton.

A second branch focused on enriching the computational power of Langton's loops: Tempesti [40] and Perrier et al. [33] demonstrated variants capable of executing attached programs, while Chou and Reggia [41] showed that populations of self-replicating loops could be adapted to (inefficiently) solve hard problems such as NP-complete satisfiability (SAT); in their design each replicant receives a partial solution to the problem that is further modified during replication and passed on to the next generation. In [39], Morita and Imai construct a 2D reversible cellular automaton capable of self-replication.

For more details, nice overviews include [7], [42] or [43].

A.1.1 Self-replicators in other systems

Computer programs Studies on quines—programs that output their own code—have been performed in [44, 45]. In [46], Ray constructed a self-replicating program using the assembly language of a virtual machine system called Tierra. He initialized a colony of programs containing this self-replicator and allowed them to evolve through interactions while competing for CPU time and memory. The programs, subject to mutations, diversified in intriguing ways, exhibiting behaviors such as symbiosis and parasitism.

In contrast to the manually designed self-replicators described above, some works explored the fascinating question of how self-replication can naturally emerge from system's dynamics, relating to the research on the origin of life. Koza [47] studied populations of Lisp programs and noticed that self-replicating programs can be found via random search, and further evolved using genetic programming. In [48], the authors considered a population of short programs written in an esoteric programming language that evolve via interacting and modifying one another. Interestingly, the authors observed an emergence of self-replicating programs.

In the program-based systems above, the *constructor* is largely externalized to the substrate's operational semantics; that is, to the ambient dynamical laws of the interpreter or virtual machine (memory allocation, instruction decoding, scheduling, divide or clone primitives). Consequently, these replicators chiefly realize transcription (copying a description), while construction or translation is effected by the host system. Said differently, the type of self-replication considered in the program-based systems entails copying the description of an organism and not the copying of the means to produce the organism, since the production mechanics is fully abstracted into the "laws of physics" of the dynamics. This stands in contrast to von Neumann– or Langton–style cellular automata, where the local transition rule is comparatively simple and domain-general, and the constructor is encoded within the spatial organization of the replicator itself.

Physical systems Interesting examples of self-replicators in physical substrates include Penrose's chains of mechanical wooden blocks [49, 50] and Jacobson's circulating trains on model railway tracks [51]; for detailed reviews see [52, 53].

A.1.2 Criteria for self-replication

The first attempt at giving a formal criterion for self-replication was presented in the seminal work by Moore [54].

Moore's criterion Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a cellular automaton with local rule f and a quiescent state (a state q satisfying $f(q, q, q, \ldots, q) = q$). Consider a configuration c which contains a pattern $\mathcal{P}$ in a finite region surrounded by the quiescent state. Then c is self-replicating if for every $k \in \mathbb{N}$ there exists a time-step t such that $F^t(c)$ contains the pattern $\mathcal{P}$ in at least k disjoint regions.

One can interpret the pattern $\mathcal{P}$ as an organism surrounded by empty space given by the quiescent state. As time progresses, the organism has to appear in a growing number of disjoint regions of the CA configuration. The definition is very natural; however, it does not capture the notion of self-replication exactly. On the one hand, it may feel too constraining: one may observe systems where the copies of $\mathcal{P}$ do not appear in the exact same form, but may be perturbed, for instance, by simple geometrical transformations. Moreover, assuming the background has to consist purely of the quiescent state may be too strict; in a similar way it proves to be too constricting when considering local universality, as exemplified by the proof of Rule 110's Turing completeness.

Indeed, there may be systems that only allow for self-replication with richer CA backgrounds. On the other hand, since Moore's criterion does not assume anything about the pattern $\mathcal{P}$, there are trivial cases that satisfy the criterion: such as a single black cell against a white background indefinitely growing in all directions.

Subsequently, more elaborate criteria have been developed, building upon Moore's initial variant. Smith [55, 56] requires the pattern $\mathcal{P}$ to be capable of universal computation (without specifying such a notion exactly). Lohn and Reggia [57] developed a variant of Moore's criterion which excludes some trivial cases by requiring the pattern $\mathcal{P}$ to change its shape as opposed to just being static; moreover, they also allow the pattern's copies to be rotated. In [58], Nehaniv and Dautenhahn extend Moore's criterion to allow for modified offsprings. In [59] Adams and Lipson discuss a graded measure of a system's capability of self-replication rather than proposing a strict binary criterion, though their application to cellular automata is illustrated only on a finite cyclic grid.

A.1.3 Self-replicators in one-dimensional cellular automata

In [55, 56] Smith proves the existence of what he calls a 1D universal self-replicator. This result, compared to our work in Appendix E, differs in a fundamental way: in the system he describes, the copying mechanism of CA structures is hardwired into the logic of the CA rule. This is in stark contrast with "genuine" self-replicators where the copying mechanism is a much higher-level process emerging from the expressive laws of the CA rules.

A.2 Local universality

The study of cellular automata as computational systems originates with von Neumann's pioneering work on self-replication. Indeed, demonstrating a CA's Turing completeness remains the most rigorous way to establish a CA's non-triviality. In order to prove a CA's local universality, one needs to show the CA can simulate another computationally universal system, most typically a universal Turing machine. There is a rich line of work constructing locally universal CAs with various properties, which we briefly review below.

Compact computation Ever since the work of Shannon [60], there has been an ongoing quest to construct small computationally universal systems. The classical line of work considers Turing machines and aims to jointly minimize their state space and alphabet size [61, 62, 63, 64, 65]. Parallel to this line of work is a series of constructions of locally universal cellular automata, progressively more compact with respect to their dimensionality, neighborhood, and number of states. These two lines of work influenced each other; in fact, some of the small universal Turing machines were used to construct compact locally universal cellular automata. In his seminal work, Banks [66] used an approach combining ideas of local and global universality: he constructed a variety of CAs that can simulate von Neumann's universal self-replicator, and are thus also computationally universal. His argument relied on implementing basic logic gates in 2D CAs. Banks was also careful enough to distinguish two cases: a 3-state, 5-neighbor CA which used finite patterns against a quiescent background, and a 2-state, 5-neighbor CA which requires an "infinite" initial configuration. A locally universal 4-state, 5-neighbor, 2D CA was constructed in [67]. Further simplifications in one and two dimensions were presented in [68, 22]. The Game of Life introduced by Conway is certainly among the most famous cellular automata and the first rule to be proven locally universal by analysis of a given rule [10, 11] as opposed to previous examples, which were carefully designed to satisfy universality. This line of work culminated in the proof of

local universality of Rule 110: a 2-state 3-neighbor 1D CA. The elaborate proof required a careful analysis of the CA's gliders and their interactions [13, 15].

An important question pertains to the space efficiency of the simulation; i.e. if system $\mathcal{B}$ (such as a CA) simulates a system $\mathcal{A}$ (such as a Turing machine), how much more time and space does $\mathcal{B}$ need to simulate t steps of $\mathcal{A}$'s computation? Some early constructions of small universal Turing machines relied on a 2-tag system (a computational system defined in D.9) simulating any Turing machine: a simulation requiring an exponential blow up in space and time. As a result, it was believed that for small universal systems, such a redundancy might be necessary. In [69], Neary and Woods showed this simulation can be realized with just a polynomial time blow-up. The authors also showed that Rule 110 can simulate a universal Turing machine in polynomial time [14], a result surprising to Cook [15].

Reversible computation Reversibility and physics-inspired constructions also played a central role. In [70] Toffoli showed that any d-dimensional CA can be simulated by a $d+1$-reversible CA, thus proving local universality for automata of dimensions 2 and higher. Subsequently, Morita and Harao [71] showed that a 1D reversible cellular automaton can simulate any reversible Turing machine, thus completing the picture for dimension one. In [72], Dubacq later showed this can be done in real time. In [28], Morita shows that any 1D CA with finite configurations (finite patterns surrounded by a quiescent state) can be simulated by a 1D reversible CA.

Computation and dynamics A closely related line of work examines the relation between computation and differentiable dynamical systems. Given a differentiable dynamical system $f : M \to M$ on a compact manifold M, we can ask what kind of computation this dynamics can instantiate, and whether it is Turing-universal. Various works [73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 30] have studied this question in various frameworks. We most closely follow and adapt the "computational dynamical systems" approach of [30].

The work [30] emphasizes that to associate the dynamics of $f : M \to M$ with a Turing machine $\mathcal{T}$ with dynamics $T : C \to C$, we require encoders $\mathcal{E} : C \to M$ and decoders $\mathcal{D} : M \to C$ such that

$$\mathcal{D} \circ f^n \circ \mathcal{E} = T^n \tag{1}$$

for all $n \geq 0$ (this generalizes to settings where there is a computational slowdown of f relative to T), where the encoder and decoder have lower computational complexity than T. While this is formalized in [30], we provide some intuition here. Suppose T is Turing-universal and we have found $\mathcal{E}, \mathcal{D}$ satisfying (1). We might be tempted to conclude that f is Turing-universal as well. However, if $\mathcal{E}, \mathcal{D}$ are themselves Turing-universal, they may 'do the work' of the Turing-universal computation in the relation (1) even if the dynamics of f is extremely simple, thus obscuring whether f exhibits computationally interesting dynamics. The basic principle, stated heuristically, is that $\mathcal{E}$ and $\mathcal{D}$ must be less computationally complex than the computation we ascribe to f (here, the computation performed by T).

In the context of differentiable dynamical systems on a compact manifold M, constraining the complexity of the encoder and decoder amounts to constraining the complexity of regions in M that encode Turing machine configurations. Regions of higher complexity are those carved out by a larger number of semi-algebraic inequalities in a controlled manner. This constraint effectively precludes the encoder and decoder from hiding computation in the extremely fine-grained, short-distance structure of the regions in M coding for Turing machine configurations.

In defining 'local universality' for CAs, we develop and adapt the work of [30]. Whereas in the differentiable dynamical systems setting on a compact manifold M the concern is that we might

encode Turing machine configurations in arbitrarily complex, small distance-scale structures of subsets of M, in the CA setting the concern is that we might encode Turing machine configurations in arbitrarily complex, large distance-scale structures of a CA configuration. In the differentiable dynamical systems setting, we grapple with arbitrarily fine structure in the continuum at finite volume, whereas in the CA setting we grapple with structure at infinite volume (where no structure exists below the distance scale of a single cell). Accordingly, our definitions in the CA setting constrain the possible complexity of infinite volume encodings of Turing machine configurations.

A.3 Global universality

A more recent line of work focuses on studying the relationship between CAs, as opposed to relating a CA's computational dynamics to a Turing machine. Indeed, in the case of local simulation, one has to grapple with the profound differences between a CA and a Turing machine: CAs are a massively parallel model of computation, operating on an infinite grid, with no predefined halting state; this is in stark contrast with the serial design of Turing machines. Informally, we say that a CA $\mathcal{B}$ globally simulates $\mathcal{A}$ if $\mathcal{B}$ can effectively reproduce any space-time diagram of $\mathcal{A}$. There are multiple ways of making precise the notion of effective reproduction of space-time diagrams which led to various notions of global simulation studied in the past. For a technical overview of the different definitions, see Section C.2. Here, we briefly summarize the main results.

The first line of work focuses on positive results: typically constructing examples of globally universal CAs with various properties. Often, the notion of global (also often called intrinsic) simulation is assumed implicitly from the construction. The seminal work of Albert and Culik [21] shows a construction of a 1D globally universal CA. This ignited the quest for the most compact, globally universal CAs [89, 90]. The already mentioned work of Toffoli [70] considers global simulation to show that any d-dimensional CA can be simulated by a $d+1$-reversible CA. This work is complemented by Hertling [27] who precisely formalizes a modern notion of global simulation to show that any d-dimensional reversible CA cannot simulate d-dimensional irreversible CAs, thus, can never be globally universal. On the other hand, Durand-Lose shows the existence of reversible automata that can simulate any other reversible automata in the same dimension [29]. Durand and Róka [23] revisit the notion of computational universality of Game of Life and show it is globally universal.

A complementary line of research focuses on the global simulation notion itself, and studying its properties, leading to a series of interesting results. Mazoyer and Rapaport [26] show the existence of infinitely many CA classes that are incomparable with respect to the relation of global simulation, and further show that no CA can be globally universal in real-time. Their results were further generalized in [91, 92, 93]. In [94], Ollinger shows that it is in general undecidable whether a CA is globally universal. For more details on these results, see Section C.2.

B Preliminaries

B.1 Cellular automata

Letting $d \in \mathbb{N}$, we call $\mathbb{Z}^d$ the *d-dimensional grid* and we call its elements the cells. For a $v \in \mathbb{Z}^d$ where $v = (v_1, \dots, v_d)$, we denote

$$\|v\| = \max\{|v_1|, \dots, |v_d|\}.$$

For a finite set S we define an *S-configuration* (sometimes just configuration) of the grid to be a mapping $c : \mathbb{Z}^d \to S$. We denote the space of all S-configurations by $S^{\mathbb{Z}^d}$, and will often write

$c \in S^{\mathbb{Z}^d}$. Sometimes, for $v \in \mathbb{Z}^d$ we write c_v instead of $c(v)$.

A *d-dimensional neighborhood* is a tuple $N = (n_1, n_2, \ldots, n_k)$ where each $n_i \in \mathbb{Z}^d$. This yields *a relative neighborhood of each cell* $v \in \mathbb{Z}^d$ to be $(v + n_1, \ldots, v + n_k)$. Now we can proceed with the definition of a CA.

Definition B.1 (Cellular automaton)**.** *A d-dimensional cellular automaton $\mathcal{A}$ is given by a neighborhood $N = (n_1, n_2, \ldots, n_k)$, a finite set of states S and a local update rule $f : S^k \to S$. This yields the CA's global update rule $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ defined for each configuration $c \in S^{\mathbb{Z}^d}$ as:*

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

The global rule updates the state of a cell v by looking at the states of cells in its relative neighborhood, and using the local rule to determine the update. This is done synchronously for all the cells in parallel. We often write $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$.

We note that a CA $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ can be defined by multiple neighborhoods and local rules. For instance, if $\mathcal{A}$ is given by neighborhood N and local rule f, we can increase the neighborhood size by adding extra cells, and adjusting the local rule so that its updates do not depend on the newly added cells. Using this approach, we can extend any neighborhood to contain exactly all cells $v \in \mathbb{Z}^d$ such that $\|v\| \leq r$ for some fixed $r \in \mathbb{N}$. If a CA's neighborhood is of such a form, we say the CA has *radius* r. In certain scenarios, this allows us to assume, without loss of generality, that each CA has radius r for some $r \in \mathbb{N}$.

Given a CA $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ and in *initial configuration* $c \in S^{\mathbb{Z}^d}$ we can iterate the CA on c to obtain a *trajectory* $c \mapsto F(c) \mapsto F^2(c) \mapsto F^3(c) \mapsto \ldots$ A nice introduction to cellular automata is for instance [2].

Visualizing CA dynamics For practical purposes, when visualizing the CA trajectories, we 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 $m = (m_1, \ldots, m_d) \in \mathbb{Z}^d$ and simply compute all the cell indices modulo m. Thus, we essentially obtain the dynamics on a discrete d-dimensional torus.

For a 1D 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 a matrix whose rows are exactly $u, F(u), F^2(u), \ldots, F^t(u)$. An example of a particular space-time diagram is shown in Fig. 5.

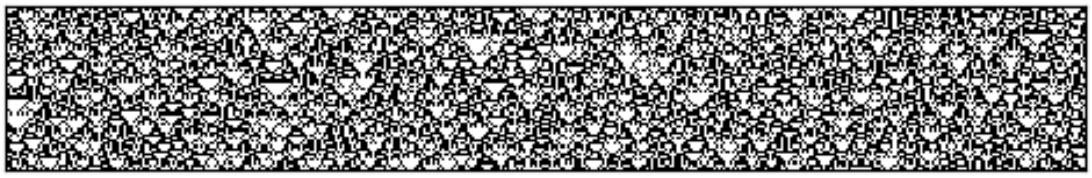


Figure 5: Space time diagram of a 1D CA with states 0 (white) and 1 (black), radius 1, and local rule $f(x, y, z) = x + z \bmod 2$ operating on a cyclic configuration of size 250. Time is progressing downwards.

Periodic configurations and shifts Let $v \in \mathbb{Z}^d$. We define the shift operator $\sigma_v : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ by $(\sigma_v(c))_w = c_{v+w}$ for each $w \in \mathbb{Z}^d$. Let $n \in \mathbb{N}$. We say that a configuration $c \in S^{\mathbb{Z}^d}$ is periodic if there exist d linearly independent vectors $v_1, \ldots, v_d$ such that $\sigma_{v_i}(c) = c$ for all $i \in \{1, \ldots, d\}$. It is clear that every CA preserves the periodicity of a configuration.

Topological dynamics We can endow the space $S^{\mathbb{Z}^d}$ with a metric as follows. We define the distance between two configurations $c_1, c_2 \in S^{\mathbb{Z}^d}$ as:

$$d(c_1, c_2) = \begin{cases} 0 & \text{if } c_1 = c_2 \\ 2^{-\min\{\|v\| \mid c_1(v) \neq c_2(v)\}} & \text{if } c_1 \neq c_2 \end{cases}.$$

Thus, two configurations are close if they agree on a large area around the origin 0. One can easily verify that this is indeed a metric that induces a topology on $S^{\mathbb{Z}^d}$, making $S^{\mathbb{Z}^d}$ a compact space with a countable base. This allows us to talk, for instance, about the continuity of maps $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$. We note that different choices of vector norms $\|\cdot\|$ can induce the same topology on $S^{\mathbb{Z}^d}$. For now, we highlight a famous result characterizing cellular automata in terms of their dynamical properties. It is quite easy to see that the global map of every CA is a continuous mapping of the topological space, and that it commutes with every shift operator. The following theorem shows also the converse, using the compactness of the topological space.

Theorem B.2 (Curtis–Hedlund–Lyndon theorem)**.** *Let S be a finite set and $d \in \mathbb{N}$. A function $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is a global rule of a CA if and only if it is continuous and commutes with every shift operator.*

For more details on topological dynamics and CAs, see for instance [95].

Subshifts of finite type Let $F \subseteq \mathbb{Z}^d$ be a finite set. By an *F-pattern* over S we understand a mapping $u : F \to S$. Each finite set of patterns L defines a *subshift of finite type (SFT)* by

$$S_L = \{c \in S^{\mathbb{Z}^d} \mid \text{no element of } L \text{ appears in c}\}.$$

It is easy to see that each SFT is a topologically closed, shift-invariant set.

Elementary cellular automata 1D cellular automata with states $\mathbf{2} = \{0, 1\}$ and a local rule with radius 1 are called elementary cellular automata (ECAs). Each local rule $f : \mathbf{2}^3 \to \mathbf{2}$ is uniquely described by its *Wolfram number* given by:

$$2^0 f(0,0,0) + 2^1 f(0,0,1) + \cdots + 2^7 f(1,1,1).$$

We will refer to each ECA by the corresponding Wolfram number of its local rule. The class of ECAs is frequently used for studying different CA properties due to its relatively small size; there are only 256 of them. Even such a small class contains CAs with complex behavior, e.g. some of them have been shown to be Turing-complete in [13]; as we will discuss in much more detail later.

Example B.3. *Fig. 5 shows the space-time dynamics of Rule 90.*

C Global simulation

C.1 Defining global simulation

Intuitively, a CA $\mathcal{B}$ globally simulates $\mathcal{A}$ if we can efficiently recover any space-time diagram of $\mathcal{A}$ from a suitable space-time diagram of $\mathcal{B}$. The correspondence between space-time diagrams of $\mathcal{A}$ and $\mathcal{B}$ are (analogously to local simulation) realized by an encoder and decoder, which will be some suitable "computationally reasonable" mappings. The configurations of the two systems are *infinite* sequences of states, which as we will see is different from the local simulation case. These

considerations will force us to only consider a fairly restricted class of encoders and decoders. The technical difficulty comes from specifying how exactly can the space-time diagrams of a CA be transformed. We say that a CA is globally universal if it can globally simulate any CA in the same dimension. Before we proceed with the formal definition, we discuss a few examples of global simulation.

C.1.1 Examples

Example C.1 (Reflections of Space-time Diagrams)**.** *Let us consider two elementary CAs, Rule 110 and Rule 124. By observing their diagrams in Fig. 6 we can notice that their local update rules are identical upon exchanging the role of left and right neighbor. This induces a relationship between the two CAs' space-time diagrams. Let* $\Theta : \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ *be the reflection of configurations over the origin. If we compare the space-time diagram of Rule 110 iterated on* $c \in \{0,1\}^{\mathbb{Z}}$ *and the space-time diagram of Rule 124 iterated from* $\Theta(c)$*, it is apparent that one is just a reflection of the other, as shown in Fig. 6. Since reflection is a computationally simple transformation, it would be natural to say that Rule 110 and Rule 124 globally simulate one another.*

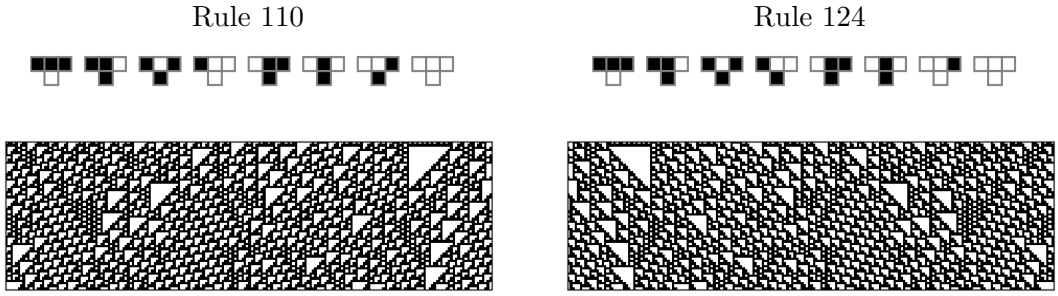


Figure 6: Rule tables and space-time diagrams of two elementary CAs.

In [21], the authors show what is probably the first construction of a globally universal CA. Specifically, they construct a 1D CA that can globally simulate any other 1D CA, however, without explicitly specifying what "globally simulate" means. Their construction follows a series of reductions, one of which is showcased in the next example.

A 1D CA is *one-way* if its local function depends only on the central cell and its right neighbor. A CA is totalistic if its state space can be embedded in $\mathbb{N}$ in such a way that its local function depends only on the sum of the states in its input.

Example C.2 (Shifts and Time Delays)**.** *For each 1D cellular automaton* $\mathcal{A}$ *with radius* $r = 1$ *there exists a one-way CA* $\mathcal{B}$ *that can simulate it [21].*

Proof. Let $\mathcal{A} = (S^{\mathbb{Z}}, F)$ be a CA with radius 1. We construct a one-way CA $\mathcal{B} = (T^{\mathbb{Z}}, G)$ as follows: $\mathcal{B}$ will simulate one iteration of $\mathcal{A}$ in two steps, first to aggregate information, second to process it. Suppose that $\mathcal{A}$ is given by the local rule $f : S^3 \to S$. We define the states of $\mathcal{B}$ as $S \cup S \times S$ and we define its local rule g as follows:

$$\begin{aligned} g(s_1, s_2) &= s_1 s_2 && \text{if } s_1, s_2 \in S \\ &= f(a, b, c) && \text{if } s_1 = ab,\ s_2 = bc \\ &\text{and is otherwise defined arbitrarily.} \end{aligned}$$

The simulation is illustrated in Fig. 7.

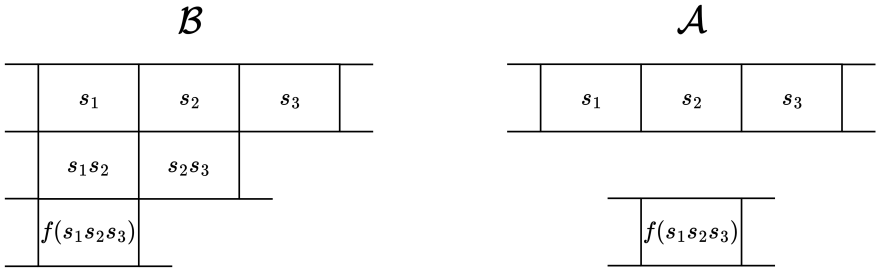


Figure 7: One-way CA simulating a CA with radius $r = 1$.

Notice that in general, the dynamics of $\mathcal{B}$ can be richer, but if the initial configuration c of $\mathcal{B}$ contains only states from S; i.e. $c \in S^{\mathbb{Z}} \subseteq (S \cup S \times S)^{\mathbb{Z}}$ then the dynamics of $\mathcal{B}$ mimics the dynamics of $\mathcal{A}$. Specifically, in order to transform any space-time diagram of $\mathcal{B}$ starting from some $c \in S^{\mathbb{Z}}$ we have to skip every second time-step since $\mathcal{B}$ takes longer to aggregate information due to its smaller neighborhood. Lastly, in order for the exact cell positions to match, we have to shift the k-th row that we did not skip by k cells to the right. We can summarize the relationship of the two CAs by relating their global rules:

$$\sigma_{-1} \circ G^2(c) = F(c) \quad \text{for all } c \in S^{\mathbb{Z}}.$$

□

In the previous two examples, whenever $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ globally simulated $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ we had that $T \supseteq S$. Clearly, if it was a necessary condition that $\mathcal{B}$ has to have a larger state space than $\mathcal{A}$ in order to simulate it, there could not exist any globally universal CA. Thus, given a CA $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ we have to be able to interpret its state space as being arbitrarily rich. This can be readily done by yet another geometrical transformation: packing. Informally, packing partitions the grid $\mathbb{Z}^d$ into larger blocks of cells; for instance for the 1D grid, the blocks could be 2 consecutive cells. This way, we can interpret every CA $\mathcal{B} = (T^{\mathbb{Z}}, G)$ as operating on a larger state space; for the case of the two consecutive cells, this would be $\mathcal{B} = (T^{\mathbb{Z}}, G) \rightsquigarrow \mathcal{B}^{\langle 2 \rangle} = \left((T^2)^{\mathbb{Z}}, G^{\langle 2 \rangle}\right)$. Before we define packing formally, we give one last example where this is utilized.

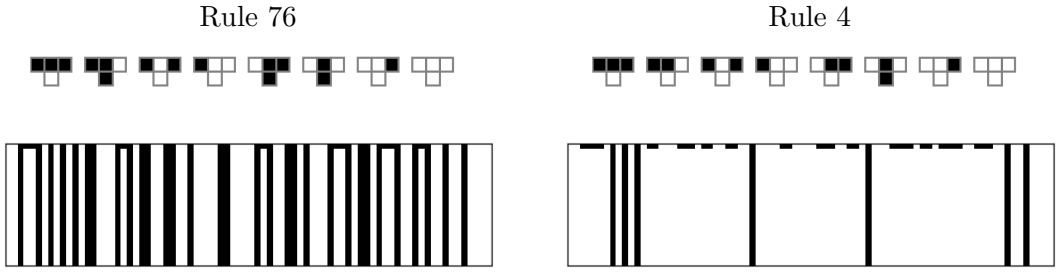


Figure 8: Rule tables and space-time diagrams of elementary CAs 76 and 4.

Example C.3 (Packing Operator)**.** *Let us consider the elementary CAs Rule 4 and Rule 76, with global functions F and G respectively (see Fig. 8).*

The dynamics of both CAs is relatively simple, since they are both idempotent; $F^2 = F$ and $G^2 = G$. Whereas Rule 4 only preserves blocks of the form 010, Rule 76 acts as identity for all neighborhoods except for 111. We will discuss that Rule 76 can simulate Rule 4. We define two local mappings:

$$\begin{aligned} e : \mathbf{2} &\to \mathbf{2}^2 & d : \mathbf{2}^2 &\to \mathbf{2} \\ 0 &\mapsto 00 & 00, 01, 10 &\mapsto 0 \\ 1 &\mapsto 11. & 11 &\mapsto 1. \end{aligned}$$

We define the encoding $\mathcal{E} : \mathbf{2}^{\mathbb{Z}} \to \mathbf{2}^{\mathbb{Z}}$ that maps each configuration, cell by cell, using the local map e and concatenating the cell tuples. Similarly, we define $\mathcal{D} : \mathbf{2}^{\mathbb{Z}} \to \mathbf{2}^{\mathbb{Z}}$ that first "partitions" each configuration into blocks of two consecutive cells, and uses the local map d to decode them. It is a simple exercise to show that for each initial configuration $c \in \mathbf{2}^{\mathbb{Z}}$ and for any number of iterations of n, it holds that $\mathcal{D} \circ G^{2n} \circ \mathcal{E}(c) = F^n(c)$; the simulation is illustrated in Fig. 9.

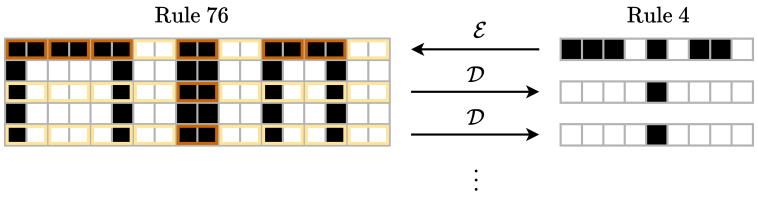


Figure 9: Rule 76 globally simulating Rule 4.

The three examples above showcase three types of geometrical transformations that were used to modify space-time diagrams of $\mathcal{B}$ to "uncover" the space-time diagrams of $\mathcal{A}$, in order to show that $\mathcal{B}$ globally simulates $\mathcal{A}$. The transformations were:

- time-delays (skipping time-steps in diagrams of $\mathcal{B}$)
- shifts (shifting spatial coordinates in diagrams of $\mathcal{B}$)
- packings (aggregating together cells into larger "supercells")

The three transformations are not arbitrary: in [91] the authors show that any geometrical transformation that maps CA space-time diagrams into CA space-time diagrams while preserving time causality is a composition of these three operators.

Let us finish this section with the formal definition of global simulation.

Definition C.4 (Global CA simulation)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ and $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ be d-dimensional cellular automata. We say that $\mathcal{B}$ globally simulates $\mathcal{A}$ if there exists a vector $v \in \mathbb{Z}^d$, a time-delay $\tau \in \mathbb{N}$, a mapping $\mathcal{E} : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$ and a partial mapping $\mathcal{D} : T^{\mathbb{Z}^d} \dashrightarrow S^{\mathbb{Z}^d}$ satisfying:*

$$\mathcal{D} \circ \sigma_v \circ G^\tau \circ \mathcal{D}^{-1}(c) = F(c) \quad \text{for all } c \in S^{\mathbb{Z}^d}, \tag{2}$$

where the mapping $\mathcal{D}$ satisfies the following three conditions:

1. *The domain of $\mathcal{D}$ is a subshift of finite type.*
2. *$\mathcal{D}$ is a continuous mapping.*
3. *There exists a full-rank matrix $M \in \mathbb{Z}^{d \times d}$ such that for all $w \in \mathbb{Z}^d$: $\sigma_w \circ \mathcal{D} = \mathcal{D} \circ \sigma_{M \cdot w}$.*

Moreover, the mapping $\mathcal{E}$ satisfies the following three conditions:

1. *$\mathcal{D} \circ \mathcal{E} = \mathrm{id}_{S^{\mathbb{Z}^d}}$.*

2. *$\mathcal{E}$ is a continuous mapping.*

3. *There exist two full-rank matrices $M_1, M_2 \in \mathbb{Z}^{d\times d}$ such that for all $w \in \mathbb{Z}^d$: $\sigma_{M_2\cdot w} \circ \mathcal{E} = \mathcal{E} \circ \sigma_{M_1\cdot w}$*

In such a case, we write $\mathcal{A} \preceq \mathcal{B}$. We say that $\mathcal{B}$ is globally universal *if it can globally simulate all d-dimensional cellular automata.*

Remark C.5 (Lower-dimensional simulation via lifting)**.** *When we say that a d-dimensional CA $\mathcal{B}$ can globally simulate* any *CA of the same or lower dimension, we handle the lower-dimensional case $d' < d$ through a standard lifting procedure: We embed $\mathbb{Z}^{d'}$ into $\mathbb{Z}^d$ via the first d' coordinates, extend any d'-dimensional configuration to be constant along the remaining $d - d'$ coordinates, and lift the local rule to ignore these extra coordinates. This transforms any d'-dimensional CA $\mathcal{A}$ into a d-dimensional CA $\mathcal{A}'$. Thus "$\mathcal{B}$ globally simulates $\mathcal{A}$" means that $\mathcal{B}$ globally simulates $\mathcal{A}'$ in the sense of Definition C.4.*

We will call $\mathcal{D}$ the decoder and $\mathcal{E}$ the encoder. In the next section, we formally introduce the packing operator and show that both the encoder and decoder are "computationally reasonable" mappings since they essentially act as cellular automata on "packed" configuration spaces. This importantly ensures that one iteration of the encoder or decoder cannot perform an arbitrarily sophisticated computation. Even though the encoder does not play a direct role in condition (2), its existence guarantees that for each initial configuration c of $\mathcal{A}$, we can efficiently recover a corresponding $c' \in \mathcal{D}^{-1}(c)$ of $\mathcal{B}$. Before we proceed with the next section, we discuss two important properties of global simulation: the relationship in (2) extends to any number of iterations of the CAs, and the relation $\mathcal{A} \preceq \mathcal{B}$ between CAs of the same dimension is a preorder. We discuss this in the following lemmas.

Lemma C.6. *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ and $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ be d-dimensional cellular automata such that $\mathcal{B}$ globally simulates $\mathcal{A}$ via a shift vector $v \in \mathbb{Z}^d$, a time-delay $\tau \in \mathbb{N}$, an encoder $\mathcal{E} : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$, and a decoder $\mathcal{D} : T^{\mathbb{Z}^d} \rightharpoonup S^{\mathbb{Z}^d}$. Then for any $n \in \mathbb{N}$ it holds that*

$$\mathcal{D} \circ (\sigma_v \circ G^\tau)^n \circ \mathcal{D}^{-1}(c) = F^n(c) \quad \text{for all } c \in S^{\mathbb{Z}^d}.$$

Proof. First, we notice that (2) implies $\sigma_v \circ G^\tau(\operatorname{dom}\mathcal{D}) \subseteq \operatorname{dom}\mathcal{D}$. Thus, for any $c \in S^{\mathbb{Z}^d}$ and for any $n \in \mathbb{N}$, $\mathcal{D} \circ (\sigma_v \circ G^\tau)^n \circ \mathcal{D}^{-1}(c)$ is non-empty. Let $c \in S^{\mathbb{Z}^d}$ and let $c' \in \operatorname{dom}(\mathcal{D})$. Since $\{c'\} \subseteq \mathcal{D}^{-1} \circ \mathcal{D}(\{c'\})$, we have that:

$$\begin{aligned}\mathcal{D} \circ (\sigma_v \circ G^\tau)^2 \circ \mathcal{D}^{-1}(c) &\subseteq \mathcal{D} \circ (\sigma_v \circ G^\tau) \circ \mathcal{D}^{-1} \circ \mathcal{D} \circ (\sigma_v \circ G^\tau) \circ \mathcal{D}^{-1}(c) \\ &\subseteq \mathcal{D} \circ (\sigma_v \circ G^\tau) \circ \mathcal{D}^{-1}(F(c)) \\ &\subseteq \{F^2(c)\}.\end{aligned}$$

Since the right-hand side is a singleton, and the left-hand side is non-empty, we obtain an equality, and thus we get that

$$\mathcal{D} \circ (\sigma_v \circ G^\tau)^2 \circ \mathcal{D}^{-1}(c) = F^2(c).$$

One can easily finish the proof by induction on n. □

As a special case, Lemma C.6 implies that

$$\mathcal{D} \circ (\sigma_v \circ G^\tau)^n \circ \mathcal{E}(c) = F^n(c) \quad \text{for all } c \in S^{\mathbb{Z}^d}.$$

This is a weaker relationship between F and G that is represented in Fig. 10 and highlights that we can compute an arbitrary iteration $F^n(c)$ using only the encoder $\mathcal{E}$, decoder $\mathcal{D}$, and iterating G.

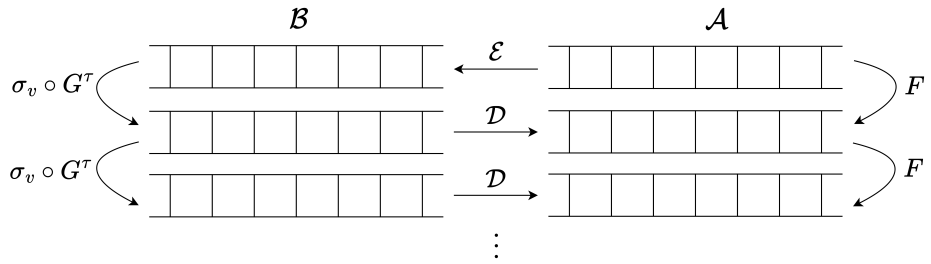


Figure 10: This diagram illustrates the relationship between $\mathcal{A}$ and $\mathcal{B}$, which holds when $\mathcal{B}$ globally simulates $\mathcal{A}$.

Lemma C.7. *The global simulation relation $\preceq$ is a preorder.*

Proof. Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$, $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$, and $\mathcal{C} = (U^{\mathbb{Z}^d}, H)$ be d-dimensional cellular automata.

Reflexivity: It is quite clear that $\mathcal{A} \preceq \mathcal{A}$ since we can simply take the shift vector to be $v = 0$, the time-delay to be $\tau = 1$ and the encoder and decoder to be the identity mappings.

Transitivity: Suppose that $\mathcal{A} \preceq \mathcal{B}$ via $v \in \mathbb{Z}^d$, $\tau \in \mathbb{N}$, $\mathcal{E} : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$ with full rank matrices M_1, M_2 and $\mathcal{D} : T^{\mathbb{Z}^d} \rightharpoonup S^{\mathbb{Z}^d}$ with a full-rank matrix M. And suppose that $\mathcal{B} \preceq \mathcal{C}$ via $v' \in \mathbb{Z}^d$, $\tau' \in \mathbb{N}$, $\mathcal{E}' : T^{\mathbb{Z}^d} \to U^{\mathbb{Z}^d}$ with full rank matrices M_1', M_2' and $\mathcal{D}' : U^{\mathbb{Z}^d} \rightharpoonup T^{\mathbb{Z}^d}$ with a full-rank matrix M'. Thus, we have that:

$$\mathcal{D} \circ \sigma_v \circ G^\tau \circ \mathcal{D}^{-1} = F$$
$$\mathcal{D}' \circ \sigma_{v'} \circ H^{\tau'} \circ \mathcal{D}'^{-1} = G\,.$$

Our goal is to show that $\mathcal{A} \preceq \mathcal{C}$. From Lemma C.6, we get that:

$$\mathcal{D}' \circ (\sigma_{v'} \circ H^{\tau'})^\tau \circ \mathcal{D}'^{-1} = G^\tau.$$

And since H commutes with any shift operator, this reduces to

$$\mathcal{D}' \circ \sigma_{v'\tau} \circ H^{\tau\tau'} \circ \mathcal{D}'^{-1} = G^\tau.$$

Using property 3 of a decoder $\mathcal{D}'$ and applying σ_v to both sides from the left, we find

$$\mathcal{D}' \circ \sigma_{M'v+v'\tau} \circ H^{\tau\tau'} \circ \mathcal{D}'^{-1} = \sigma_v \circ G^\tau$$

which finally yields

$$\mathcal{D} \circ \mathcal{D}' \circ \sigma_{M'v+v'\tau} \circ H^{\tau\tau'} \circ \mathcal{D}'^{-1} \circ \mathcal{D}^{-1} = \mathcal{D} \circ \sigma_v \circ G^\tau \circ \mathcal{D}^{-1} = F.$$

Thus, for the decoder $\tilde{\mathcal{D}} = \mathcal{D} \circ \mathcal{D}'$, shift vector $\tilde{v} = M'v + v'\tau$, and time delay $\tilde{\tau} = \tau\tau'$ it holds that $\tilde{\mathcal{D}} \circ \sigma_{\tilde{v}} \circ G^{\tilde{\tau}} \circ \tilde{\mathcal{D}}^{-1} = F$. Naturally, we define the corresponding encoder as $\tilde{\mathcal{E}} = \mathcal{E}' \circ \mathcal{E}$. Lastly, we have to check that $\tilde{\mathcal{E}}$ and $\tilde{\mathcal{D}}$ satisfy all conditions of Definition C.4. It is clear that both $\tilde{\mathcal{E}}$ and $\tilde{D}$ are continuous and that $\tilde{\mathcal{D}} \circ \tilde{\mathcal{E}} = \mathrm{id}_{S^{\mathbb{Z}^d}}$. Next, we show that $\tilde{E}$ commutes with certain shifts determined by a pair of full rank matrices. This would clearly be true if $M_2 \cdot w \in \operatorname{rng} M_1'$ for all $w \in \mathbb{Z}^d$. This does not hold in general, but one can notice that $\det(M_1')M_2 \cdot w \in \operatorname{rng} M_1'$ for all $w \in \mathbb{Z}^d$. Let A be such that $M_1'A = \det(M_1')M_2$. Then, for any $w \in \mathbb{Z}^d$:

$$\begin{aligned}\tilde{\mathcal{E}} \circ \sigma_{\det(M_1')M_1 w} &= \mathcal{E}' \circ \mathcal{E} \circ \sigma_{\det(M_1')M_1 w} = \mathcal{E}' \circ \sigma_{\det(M_1')M_2 w} \circ \mathcal{E} \\ &= \mathcal{E}' \circ \sigma_{M_1'Aw} \circ \mathcal{E} = \sigma_{M_2'Aw} \circ \mathcal{E}' \circ \mathcal{E} = \sigma_{M_2'Aw} \circ \tilde{\mathcal{E}}.\end{aligned}$$

Therefore, $\tilde{\mathcal{E}}$ satisfies condition 3 of an encoder for matrices $\tilde{M}_1 = \det(M_1')M_1$ and $\tilde{M}_2 = M_2'A$. Verifying condition 3 for the decoder is quite straightforward. Lastly, we have to check that $\mathrm{dom}\,(\tilde{\mathcal{D}})$ is a subshift of finite type:

$$\mathrm{dom}\,(\mathcal{D} \circ \mathcal{D}') = \{x \in \mathrm{dom}\,(\mathcal{D}') \mid \mathcal{D}'(x) \in \mathrm{dom}\,(\mathcal{D})\} = \mathrm{dom}\,(\mathcal{D}') \cap \mathcal{D}'^{-1}(\mathrm{dom}\,(\mathcal{D})).$$

It is easy to check that the intersection of two SFTs is again a SFT. Furthermore, from the next section, it will be apparent that since $\mathcal{D}'$ is continuous and commutes with certain shifts, it can be viewed as a "cellular transformaton", and thus $\mathcal{D}'^{-1}$ maps SFTs to SFTs. □

C.1.2 Packing Operator

Packings are a family of geometrical transformations of CA configurations, introduced in full generality in [91], that form a key ingredient of any notion of CA global simulation. Intuitively, given a pattern $\mathcal{P} \subset \mathbb{Z}^d$ of size k which induces a tiling of $\mathbb{Z}^d$, the packing operator packs together the cells in each tile into a larger "supercell"; transforming configurations from $S^{\mathbb{Z}^d}$ to $(S^k)^{\mathbb{Z}^d}$.

Formally, we call any finite subset $\mathcal{P} \subset \mathbb{Z}^d$ a *pattern*. Given a vector $v \in \mathbb{Z}^d$, the translation of $\mathcal{P}$ by v is $\mathcal{P} + v = \{p + v \mid p \in \mathcal{P}\}$. Given a sequence $V = (v_1, \ldots, v_d)$ of d linearly independent vectors from $\mathbb{Z}^d$, we write $M_V = (v_1|v_2|\cdots|v_d) \in \mathbb{Z}^{d\times d}$ to denote the matrix whose columns are given by V. We call $(\mathcal{P}, V)$ *a tiling* of $\mathbb{Z}^d$ if the set $\{\mathcal{P} + M_V \cdot z \mid z \in \mathbb{Z}^d\}$ is a partition of $\mathbb{Z}^d$. We can notice that V induces an equivalence relation on $\mathbb{Z}^d$ defined by $z \equiv_V z'$ if and only if $z - z' \in M_v \cdot x$ for some $x \in \mathbb{Z}^d$. Clearly, $\equiv_V$ has only finitely many equivalence classes, their number equal to $\det(M_v)$. We can observe that $(\mathcal{P}, V)$ is a tiling of $\mathbb{Z}^d$ if and only if $\mathcal{P}$ contains exactly one element from each equivalence class induced by V. An example of a tiling of $\mathbb{Z}^2$ is illustrated in Fig. 11.

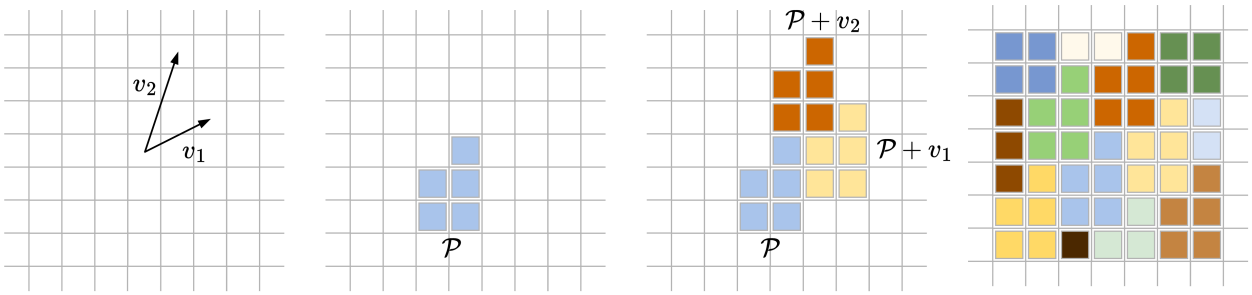


Figure 11: (left) Vectors $v_1 = (2, 1)$, $v_2 = (1, 3)$ spanning a sublattice in $\mathbb{Z}^2$. (middle-left) The pattern $P = \{(0, 0), (1, 0), (0, 1), (1, 1), (1, 2)\}$. (middle-right) Translation of $\mathcal{P}$ by v_1 and by v_2. (right) Tiling of $\mathbb{Z}^2$ induced by $(\mathcal{P}, (v_1, v_2))$.

Let $(\mathcal{P}, V)$ be a tiling of $\mathbb{Z}^d$ with $|\mathcal{P}| = m$, and let us fix the order of the elements of $\mathcal{P}$; $\mathcal{P} = (p_1, \ldots, p_m)$. The tiling $(\mathcal{P}, V)$ induces a *packing operator* $o^{\langle \mathcal{P},V\rangle} : S^{\mathbb{Z}^d} \to (S^m)^{\mathbb{Z}^d}$ for any finite set S, defined as follows:

$$o^{\langle \mathcal{P},V\rangle}(c)(z) = \big(c(p_1 + M_v \cdot z), c(p_2 + M_v \cdot z), \ldots, c(p_m + M_v \cdot z)\big) \in S^m, \text{for } c \in S^{\mathbb{Z}^d} \text{ and } z \in \mathbb{Z}^d.$$

The packing operator is illustrated in Fig. 12.

Given a cellular automaton $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ and a packing $(\mathcal{P}, V)$ of $\mathbb{Z}^d$ with $|\mathcal{P}| = m$, we can transform $\mathcal{B}$ into $\mathcal{B}^{\langle \mathcal{P},V\rangle} = \Big((T^m)^{\mathbb{Z}^d}, G^{\langle \mathcal{P},V\rangle}\Big)$ where $G^{\langle \mathcal{P},V\rangle} = o^{\langle \mathcal{P},V\rangle} \circ G \circ \big(o^{\langle \mathcal{P},V\rangle}\big)^{-1}$. This provides a natural way to view a given CA as operating on a larger state space. Below, we discuss that

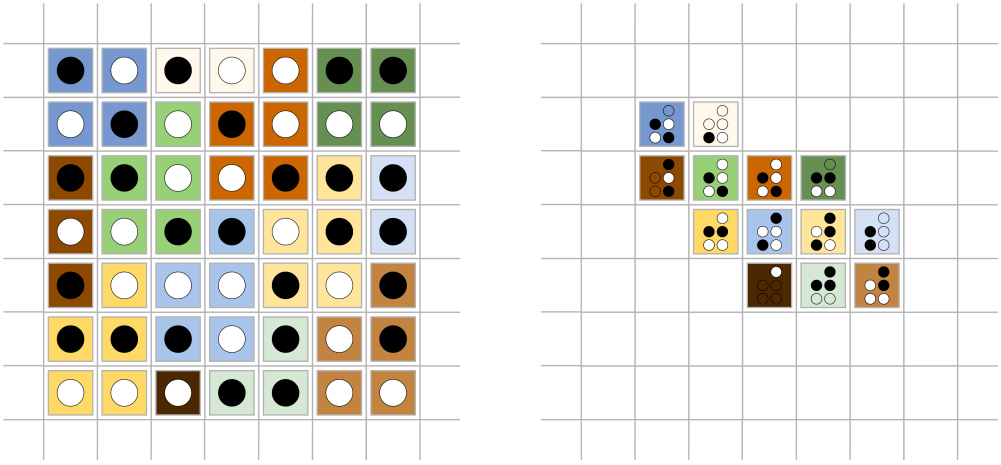


Figure 12: The tiling $(\mathcal{P}, V)$ from Fig. 11 induces a packing operator. This figure illustrates the case of $o^{\langle \mathcal{P}, V \rangle} : \mathbf{2}^{\mathbb{Z}^2} \to (\mathbf{2}^5)^{\mathbb{Z}^2}$. (Left) Depiction of a configuration $c \in \mathbf{2}^{\mathbb{Z}^2}$. White and black circles illustrate the states 0 and 1; we keep the background coloring to show the space tiling. (Right) We display the outcome $o^{\langle \mathcal{P}, V \rangle}(c)$. The states are aggregated by groups of 5.

a decoder and encoder witnessing a global simulation $\mathcal{A} \preceq \mathcal{B}$ are essentially CAs operating on a packed grid space. Since their input and output state spaces differ, we generalize the notion of CAs below.

Definition C.8 (Cellular transformaton)**.** *Let S and T be finite sets and $d \in \mathbb{N}$. We say that $F : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$ is a* cellular transformaton *if there exists a neighborhood $(n_1, n_2, \dots, n_k) \subseteq \left(\mathbb{Z}^d\right)^k$ and a local rule $f : S^k \to T$ such that for each $v \in \mathbb{Z}^d$ and for each $c \in S^{\mathbb{Z}^d}$:*

$$F(c)_v = f\left(c_{v+n_1}, \dots, c_{v+n_k}\right).$$

Cellular transformata have been studied in the context of symbolic dynamics under the name of *sliding block codes* [96, 97]. Following the exact same proof strategy as in the Curtis-Hedlund-Lyndon Theorem B.2, we obtain an analogous result for cellular transformata:

Lemma C.9 (Curtis–Hedlund–Lyndon theorem for cellular transformata)**.** *Let S, T be finite sets and $d \in \mathbb{N}$. Let $F : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$. Then the following conditions are equivalent:*

(i) F is continuous and commutes with all shift operators.

(ii) F is a cellular transformaton.

Now we can proceed with the characterization of the encoders and decoders, using a simple generalization of the Curtis-Hedlund-Lyndon Theorem B.2.

Theorem C.10 (Packing version of the Curtis–Hedlund–Lyndon theorem for cellular transformata)**.** *Let S, T be finite sets and $V = (v_1, \dots, v_d)$, $W = (w_1, \dots, w_d)$ two sequences of d linearly independent vectors in $\mathbb{Z}^d$; we denote their corresponding matrices by M_V and M_W. Let $(\mathcal{P}, V)$ and $(\mathcal{Q}, W)$ be two packings of $\mathbb{Z}^d$. Then, the following two conditions are equivalent:*

(i) F is continuous and for each $v \in \mathbb{Z}^d$ satisfies $F \circ \sigma_{M_V \cdot v} = \sigma_{M_W \cdot v} \circ F$.

(ii) $G = o^{\langle \mathcal{Q}, W \rangle} \circ F \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$ is a cellular transformaton.

Proof. Here we provide a sketch of the proof. $(i) \implies (ii)$: It suffices to verify that G is continuous and commutes with all shifts on the "packed space". The continuity is clear since all packing operators are continuous. Let $v \in \mathbb{Z}^d$. Then:

$$\begin{aligned} G \circ \sigma_v &= o^{\langle \mathcal{Q}, W \rangle} \circ F \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1} \circ \sigma_v \\ &= o^{\langle \mathcal{Q}, W \rangle} \circ F \circ \sigma_{M_V \cdot v} \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1} \\ &= o^{\langle \mathcal{Q}, W \rangle} \circ \sigma_{M_W \cdot v} \circ F \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1} \\ &= \sigma_v o^{\langle \mathcal{Q}, W \rangle} \circ F \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1} = \sigma_v \circ G. \end{aligned}$$

Thus, using Lemma C.9, $G : (S^{|\mathcal{P}|})^{\mathbb{Z}^d} \to (T^{|\mathcal{Q}|})^{\mathbb{Z}^d}$ is a cellular transformaton.
$(ii) \implies (i)$: Again, we use Lemma C.9 to see that G commutes with all shifts and is continuous. Utilizing similar arguments as above, we get (i). □

Hence, we can interpret each decoder witnessing a CA global simulation $\mathcal{A} \preceq \mathcal{B}$ as an operator that packs the configurations of $\mathcal{B}$ and transforms them into configurations of $\mathcal{A}$. Similarly, $\mathcal{E}$ packs configurations of $\mathcal{A}$, transforms them into packed configurations of $\mathcal{B}$, and unpacks those. We illustrate this by the following example, again coming from the work of Culik and Yu [21]. We say a CA is totalistic if its local rule is invariant to any permutation of its inputs.

Example C.11 (Totalistic simulation [21])**.** *For each 1D cellular automaton $\mathcal{A} = (S^{\mathbb{Z}}, F)$ with radius $r = 1$ there exists a totalistic CA $\mathcal{B}$ also with radius 1 that globally simulates it.*

Proof. We simply encode the relative positions of cells into the states of $\mathcal{B}$ by the encoder. Suppose $\mathcal{A}$ has states $\{s_1, s_2, \ldots, s_n\}$ that we identify with numbers $1, \ldots, n$ and let $B = n+1$. The encoding is based on using the numbers 10, 100, and 1000 in B-ary representation, as seen in Fig. 13.

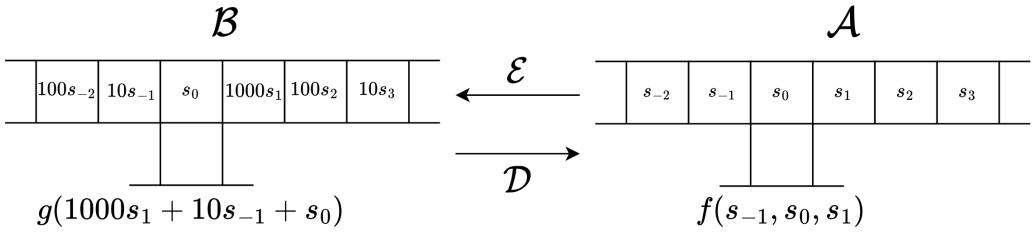


Figure 13: Encoded configuration of a totalistic CA simulating a CA with radius $r = 1$.

It is clear that the sum of any three consecutive cells will contain exactly one 0 that determines the position of the three states in the neighborhood. This definition shows that the encoder requires information about the positions of the cells. The decoder simply "forgets" this additional information and sends each 4-tuple of cells $s_0, 1000s_1, 100s_2, 10s_3 \mapsto s_0s_1s_2s_3$.

One can easily see that the encoder packs the 1D grid into blocks of four consecutive cells, and transforms them via a cellular transformaton (with radius one), to unpack them again. Formally, $\mathcal{E}$ satisfies for any $v \in \mathbb{Z}$ that $\mathcal{E} \circ \sigma_{4v} = \sigma_{4v} \circ \mathcal{E}$. The domain of decoder $\mathcal{D}$ contains exactly all valid encodings of configurations, i.e. $\mathrm{dom}(\mathcal{D}) = \mathcal{E}(S^{\mathbb{Z}})$. $\mathcal{D}$ does not need to pack the configuration space of $\mathcal{B}$ and can be simply implemented as a cellular transformaton with radius 1 which forgets the positional encoding.

Since for each $c \in S^{\mathbb{Z}}$, $\mathcal{D}^{-1}(c) = \{\mathcal{E}(c)\}$, it is easy to see that $\mathcal{D} \circ G \circ \mathcal{D}^{-1} = F$. □

It is now an easy exercise to return to the examples we gave at the beginning, and to show they "comply" with Definition C.4 of global simulation.

C.2 Related work

Different variants of global simulation and their properties have been studied in the past; typically under the name of *intrinsic simulation*. There are at least three distinct lines of work, represented by [26], [91], and [93] which study notions that are similar in spirit, yet differ in the details. As there is no comparison of these works in the literature, we fill this gap by providing a detailed overview, summarized in Table 1.

For this whole section, we consider $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ and $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ to be cellular automata in dimension $d \in \mathbb{N}$. All of the variants of CA simulation can be formulated in the same general framework:

$\mathcal{B}$ simulates $\mathcal{A}$ if there exists a certain transformation $\tilde{G}$ of G, a transformation $\tilde{F}$ of F and a decoder $\mathcal{D}$ some computationally feasible partial mapping translating between configurations of $\mathcal{B}$ and $\mathcal{A}$ such that

$$\mathcal{D} \circ \tilde{G} \circ \mathcal{D}^{-1}(c) = \tilde{F}(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

Global simulation For global simulation, we don't allow for transformations of F, thus $\tilde{F} = F$. For G, we have $\tilde{G} = o^{\langle \mathcal{P}, V \rangle} \circ \sigma_v \circ G^\tau \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$ for some shift vector $v \in \mathbb{Z}^d$, some time-delay $\tau \in \mathbb{N}$, and some tiling $(\mathcal{P}, V)$ of $\mathbb{Z}^d$.

The decoder is a cellular transformaton $\mathcal{D} : (T^m)^{\mathbb{Z}^d} \rightharpoonup S^{\mathbb{Z}^d}$ whose domain is a subshift of finite type and such that we can efficiently find preimages of $\mathcal{D}$ via an injective cellular transformaton $\mathcal{E}$.

Grouping In their seminal work [26], Mazoyer and Rapaport study the properties of what they call a grouping operation on 1D CAs. In our notation, $\tilde{G} = o^{\langle \mathcal{P}, V \rangle} \circ G^m \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$ for a simple tiling $(\mathcal{P}, V)$ where $\mathcal{P}$ is a rectangle of size m and $V = (m)$. Similarly, they allow to transform F into $\tilde{F} = o^{\langle \mathcal{Q}, W \rangle} \circ F^n \circ \left(o^{\langle \mathcal{Q}, W \rangle}\right)^{-1}$ again with a rectangular tiling $(\mathcal{Q}, W)$ of size n. Notice that here, the shift vector is 0 and the time-delay is exactly the size of the tiling. This enforces the CAs to process information "in real time".

The decoder $\mathcal{D}$ is an injective cellular transformaton $\mathcal{D} : (T^m)^{\mathbb{Z}} \rightharpoonup (S^n)^{\mathbb{Z}}$ with radius 0 and a domain $(T')^{\mathbb{Z}}$ where $T' \subseteq T^m$. The authors obtain deep results about their notion of CA simulation in dimension 1 which include the following:

- Assuming the real-time simulation, i.e. time-delay of $\mathcal{B}$ is equal to the tiling's size, the CA simulation admits no universal CA.
- There exists a countable number of incomparable CA classes with respect to the relation induced by the simulation [98].
- The general problem of whether $\mathcal{B}$ simulates $\mathcal{A}$ is undecidable.

Bulking In the subsequent series of very relevant works [91] and [92] by Delorme, Mazoyer, Ollinger, and Theyssier, the authors consider a generalization of the grouping operation which they call bulking. For $\mathcal{B}$ and $\mathcal{A}$ in arbitrary dimension $d \in \mathbb{N}$ they consider $\tilde{G} = o^{\langle \mathcal{P}, V \rangle} \circ \sigma_v \circ G^\tau \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$ where $v \in \mathbb{Z}^d$, $\tau \in \mathbb{N}$ and $(\mathcal{P}, V)$ is a rectangular tiling of $\mathbb{Z}^d$; i.e. $\mathcal{P}$ is a rectangle, and V consists of canonical vectors, multiplied by scalars and possibly permuted. Similarly, they consider $\tilde{F} = o^{\langle \mathcal{Q}, W \rangle} \circ \sigma_w \circ F^{\tau'} \circ \left(o^{\langle \mathcal{Q}, W \rangle}\right)^{-1}$ where $(\mathcal{Q}, W)$ is again a rectangular packing of $\mathbb{Z}^d$. Lastly, in [92],

the most general decoder the authors consider is a cellular transformaton $\mathcal{D} : T^{|\mathcal{P}|^{\mathbb{Z}^d}} \rightharpoonup S^{|\mathcal{Q}|^{\mathbb{Z}^d}}$ with radius 0 whose domain is a again a set $(T')^{\mathbb{Z}^d}$ where $T' \subseteq T^{|\mathcal{P}|}$. Among other results, in [91], the authors importantly show that any geometrical transformation that maps CA space-time diagrams into CA space-time diagrams while preserving time causality is a composition of a (general) packing, time-delay, and a shift. For more details, see also the thesis of Ollinger [99] or Theyssier [100].

Algebraic simulation In [93], Hudcová and Krásenský formulate the notion of CA simulation in algebraic language which helps them use some deeper results from universal algebra to show that certain CA classes are limited in terms of what they can simulate. Concretely, they focus on 1D CAs and consider $\tilde{F} = F$ and $\tilde{G} = \tilde{G}_1 \times \tilde{G}_2 \times \cdots \tilde{G}_k$ where $\times$ denotes the standard CA product, and each $\tilde{G}_i = o^{\langle \mathcal{P}_i, V_i \rangle} \circ G^{m_i} \circ \left(o^{\langle \mathcal{P}_i, V_i \rangle}\right)^{-1}$ where $(\mathcal{P}_i, V_i)$ is a rectangular packing of size m_i. The authors consider the decoder to be a cellular transformaton with radius 0 defined on a subshift of finite type.

The introduction of products has an algebraic motivation since for every CA $\mathcal{B}$ the set $\{\mathcal{A} \mid \mathcal{B}$ can simulate $\mathcal{A}\}$ has a nice structure: it forms a pseudovariety. The authors show that (almost) any affine CA (local rule is an affine mapping of vector spaces) or Abelian CA (local rule is a homomorphism of an Abelian group) can only simulate CAs with the same property, thus, in particular they cannot be globally universal. A widely studied affine CA is for instance the elementary CA Rule 90. More details can be found in Hudcová's thesis [101].

We present an overview of the variety of related notions in Table 1.

Global Simulation Framework

$\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ simulates $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ if $\mathcal{D} \circ \tilde{G} \circ \mathcal{D}^{-1} = \tilde{F}$ for some decoder $\mathcal{D}$

Sim. Type	Dim.	$\tilde{\mathbf{G}}$	$\tilde{\mathbf{F}}$	decoder $\mathcal{D}$
grouping [26]	$d = 1$	$o^{\langle \mathcal{P}, V \rangle} \circ G^{\lvert\mathcal{P}\rvert} \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$, $(\mathcal{P}, V)$ a rectangular tiling	$o^{\langle \mathcal{Q}, W \rangle} \circ F^{\lvert\mathcal{Q}\rvert} \circ \left(o^{\langle \mathcal{Q}, W \rangle}\right)^{-1}$, $(\mathcal{Q}, W)$ a rectangular tiling	injective CT* with radius 0
bulking [91, 92]	$d \in \mathbb{N}$	$o^{\langle \mathcal{P}, V \rangle} \circ \sigma_v \circ G^{\tau} \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$, $(\mathcal{P}, V)$ a rectangular tiling	$o^{\langle \mathcal{Q}, W \rangle} \circ \sigma_w \circ F^{\tau'} \circ \left(o^{\langle \mathcal{Q}, W \rangle}\right)^{-1}$, $(\mathcal{Q}, W)$ a rectangular tiling	CT* with radius 0
algebraic [93]	$d = 1$	$\tilde{G}_1 \times \tilde{G}_2 \times \cdots \times \tilde{G}_k$, each $\tilde{G}_i$ as in grouping	F	CT* with radius 0
global	$d \in \mathbb{N}$	$o^{\langle \mathcal{P}, V \rangle} \circ \sigma_v \circ G^{\tau} \circ \left(o^{\langle \mathcal{P}, V \rangle}\right)^{-1}$, $(\mathcal{P}, V)$ a general tiling	F	CT* with radius r

Table 1: Global Simulation: Overview of Related Works.
*Here, CT stands for a cellular transformaton with a local rule being a partial mapping.

D Local simulation

Local simulation refers to the ability of a cellular automaton to mimic the computation of a given Turing machine. Locality refers to the fact that the Turing machine's tape is, at any given moment, just a finite set of symbols, which is contained in a bounded region of the CA's configuration via a local encoding mapping. The notion of a CA simulating a Turing machine is simple to introduce conceptually, yet in order to obtain a concrete formal definition, a myriad of technical details need to be specified. Precisely specifying such details appears to be more challenging than what could initially be expected; as Ollinger writes in his review Universalities in Cellular Automata [16]:

"... (this) leads to the problem of heterogeneous simulation: classical models of computation have inputs, step function, halting condition, and output. Cellular automata have no halting condition and no output. As pointed out by Durand and Róka [23], this leads to very tricky encoding problems: their own attempt at a Turing universality based on this criterion as encoding flaw permitting us counterintuitively to consider very simple cellular automata as universal. Turing universality of dynamical systems in general and cellular automata in particular has been further discussed by Delvenne et al. [102] and Sutner [103]. None of the proposed definitions is completely convincing so far, so it has been chosen on purpose not to provide the reader with yet another weak formal definition."

It is surprising that despite the rich line of works studying Turing universality within CAs, no formal definition succeeded in grasping this notion in a satisfying manner. The goal of this section is to propose a formal definition of CA Turing completeness that does not lead to pathological examples of trivial yet Turing-complete automata, and at the same time is general enough to agree with all the constructions of Turing-complete cellular automata from the literature. Before delving into the details, we first provide the conceptual overview of local simulation.

D.1 Defining local simulation

Local simulation: 1st level of resolution Very roughly, we say that a CA $\mathcal{A}$ simulates a Turing machine $\mathcal{T}$ if, for every initial tape configuration c of $\mathcal{T}$ there exists a corresponding initial configuration c' of $\mathcal{A}$ such that the space-time diagram of $\mathcal{A}$ contains the snapshots of all computation steps of $\mathcal{T}$ ($\mathcal{T}$ need not be a Turing machine that always halts). Further, we require that the function mapping $c \mapsto c'$ is a computationally reasonable mapping. Similarly, we require that for every time-step $t \in \mathbb{N}$ we can recover t steps of time evolution of c, denoted by $T^t(c)$, from the space-time diagram of $\mathcal{A}$ in a computationally feasible way.

Definition D.1 (Turing machine)**.** *A Turing machine $\mathcal{T}$ is given by a triple (Q, Σ, δ) where:*

1. *Q is a finite set of states, with a designated $q_0 \in Q$ the* initial state *and $q_{\text{halt}} \in Q$ the* halting state*.*
2. *Σ is a finite set representing the* tape alphabet*, with a designated blank symbol $\sqcup$*
3. *$\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, S, R\}$ is the* transition function*.*

We define the configuration space *of $\mathcal{T}$ to be the set $C_{\mathcal{T}} = \Sigma^* \times Q \times \Sigma^*$. A configuration $c \in C_{\mathcal{T}}$, $c = s_1 \cdots s_{m-1} q s_m \cdots s_n$, defines the Turing machine's tape content $s_1 \cdots s_n \in \Sigma^n$ (with all the other symbols on the bi-infinite tape being blank), the current state to be $q \in Q$ and the Turing machine head currently reading s_m. Thus, we can interpret $\mathcal{T}$ as a function $T : C_{\mathcal{T}} \to C_{\mathcal{T}}$ as follows. Let $\delta(q, s_m) = (q', s'_m, a)$ for $a \in \{\mathrm{L}, \mathrm{S}, \mathrm{R}\}$, then*

$$T(c) = \begin{cases} s_1 \cdots s_{m-2} q' s_{m-1} s'_m \cdots s_n & \text{if } a = \mathrm{L} \\ s_1 \cdots s_{m-1} q' s'_m \cdots s_n & \text{if } a = \mathrm{S} \\ s_1 \cdots s_{m-1} s'_m q' s_{m+1} \cdots s_n & \text{if } a = \mathrm{R}. \end{cases}$$

We will identify each Turing machine $\mathcal{T}$ with the dynamical system $T : C_{\mathcal{T}} \to C_{\mathcal{T}}$.

Local simulation: 2nd level of resolution Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a d-dimensional cellular automaton and $\mathcal{T} = (C_{\mathcal{T}}, T)$ a Turing machine. We say that $\mathcal{A}$ simulates $\mathcal{T}$ if there exists:

1. an *encoder* $\mathcal{E} : C_{\mathcal{T}} \to S^{\mathbb{Z}^d}$

2. a *decoder* $\mathcal{D} : S^{\mathbb{Z}^d} \to C_{\mathcal{T}}$
3. a delay function $\tau : S^{\mathbb{Z}^d} \to C_{\mathcal{T}}$

Such that for all $c \in C_{\mathcal{T}}$ and for all time-steps $t \in \mathbb{N}$ it holds that:

$$\mathcal{D} \circ (F^\tau)^t \circ \mathcal{E}(c) = T^t(c) \tag{3}$$

where $F^\tau : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is defined as $F^\tau(c) = F^{\tau(c)}(c)$ for any $c \in S^{\mathbb{Z}^d}$. The relationship between $\mathcal{A}$ and $\mathcal{T}$ captured in equation (3) is illustrated in Fig. 14.

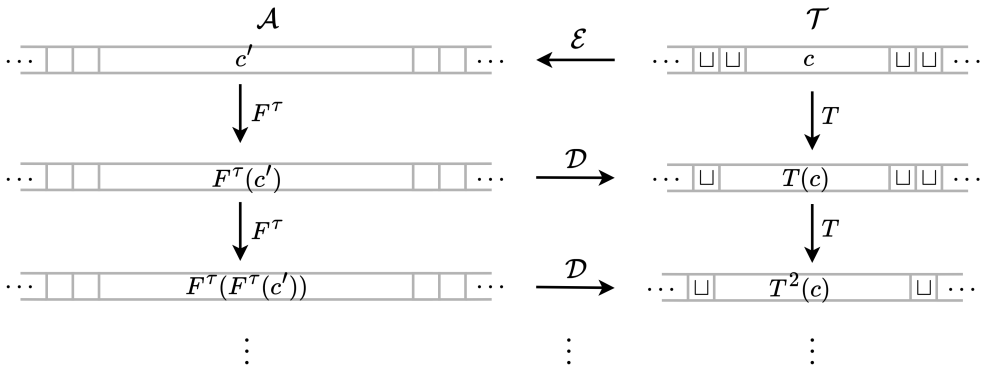


Figure 14: A general scheme illustrating the concept of local simulation: cellular automaton $\mathcal{A}$ simulates the Turing machine $\mathcal{T}$ via an encoder $\mathcal{E}$, decoder $\mathcal{D}$, and a delay function τ.

Crucially, the mappings $\mathcal{E}, \mathcal{D}$, and τ have to be "computationally reasonable". Intuitively, we want to prevent cases where $\mathcal{E}$, $\mathcal{D}$, or τ could perform an arbitrarily sophisticated computation in a single iteration. However, specifying what "computationally reasonable" means exactly is where the technical challenge lies. First, the mappings relate finite configurations of $\mathcal{T}$ with infinite configurations of $\mathcal{A}$. To deal with this, it is natural to require the following. Let $c \in C_{\mathcal{T}}$ with $|c| = n$ and let $\mathcal{D}(c') = c$.

1. $\mathcal{E}$ maps c to a finite region of size proportional to n, centred around 0. The configuration outside this region is initialized with some "admissible background" (a natural choice is, for instance, the quiescent state if the CA has one).
2. When decoding c', $\mathcal{D}$ has access only to a finite region centred around 0 whose size is proportional to n.
3. Similarly to $\mathcal{D}$, when τ computes the time-delay for c', it only has access to a finite region centred around 0 whose size is proportional to n.

Thus, we have reduced the mappings $\mathcal{E}, \mathcal{D}$ and τ to ones between finite configurations. As such, we can require them to be realized by computable functions of bounded complexity (concretely, we will bound their time complexity by some computable function $\beta(n)$).

Before we proceed with the formal definition, we discuss two examples of what can go wrong, if we impose no complexity restrictions on the mappings $\mathcal{D}, \mathcal{E}$, and τ.

Example D.2 (Unrestricted Decoder)**.** *We adapt an example from [30] to show that when the decoder's computational complexity is not restricted, even a trivial CA can simulate a universal Turing machine.*

Let $\mathcal{T}_{\text{univ}} = (C_{\mathcal{T}_{\text{univ}}}, T_{\text{univ}})$ be a universal Turing machine given by (Q, Σ, δ). We define a trivial 1D CA $\mathcal{A}$ with states $S = Q \cup \Sigma \cup \{0, 1\}$ and radius 1. The local rule $f : S^3 \to S$ acts as the identity of the central cell except for the following three cases:

$$f(0q_0\sqcup) = 1, \quad f(1q_0\sqcup) = 1, \quad f(q_0 \sqcup \sqcup) = q_0.$$

Let $c \in C_{\mathcal{T}_{\text{univ}}}$ with length n, we define $\mathcal{E}(c)\big|_{[0,n+2]} = c\,0\,q_0$ and to be $\sqcup$ everywhere else. We define τ to be a constant 1, and we define the decoder

$$\mathcal{D}(\cdots \sqcup \sqcup c\,0\,\underbrace{11\cdots1}_{k}\,q_0 \sqcup \sqcup \cdots) \coloneqq T^k_{\text{univ}}(c).$$

Now it is easy to see that if we iterate $\mathcal{A}$ on $\mathcal{E}(c)$, it keeps all the information about c and q_0 intact and simply tracks the number of iterations by a growing sequence of 1s. Therefore, we see that $\mathcal{A}$ simulates $\mathcal{T}_{\text{univ}}$ as it satisfies equation (3).

Example D.3 (Unrestricted Encoder)**.** *We adapt an example from [23] to show that if we impose no complexity restrictions on the encoder and delay function, even a trivial CA can simulate a universal Turing machine.*

Let $\mathcal{T}_{\text{univ}} = (C_{\mathcal{T}_{\text{univ}}}, T_{\text{univ}})$ be a universal Turing machine given by (Q, Σ, δ). We define a trivial 1D CA $\mathcal{A}$ with states $S = Q \cup \Sigma$ and radius 1, which simply shifts every configuration by one cell to the left. Let $c \in C_{\mathcal{T}_{\text{univ}}}$, we define the encoder to already "precompute" all iterations of $\mathcal{T}_{\text{univ}}$ on c:

$$\mathcal{E}(c) \coloneqq \cdots \sqcup \sqcup . c \sqcup T_{\text{univ}}(c) \sqcup T^2_{\text{univ}}(c) \sqcup \cdots$$

where c starts at position 0. Whenever we have a CA configuration of the form $\cdots \sqcup \sqcup . u_1 \sqcup u_2 \sqcup \cdots$ for $u_1, u_2, \ldots \in S^*$ we define $\tau(\cdots \sqcup \sqcup . u_1 \sqcup u_2 \sqcup \cdots) = |u_1| + 1$. Lastly, we define the decoder $\mathcal{D}(\cdots \sqcup \sqcup . u_1 \sqcup u_2 \sqcup \cdots) = u_1$. Again, it is easy to check that $\mathcal{A}$ simulates $\mathcal{T}_{\text{univ}}$ via $\mathcal{E}, \mathcal{D}$, and τ while all the "computation" is hidden in the encoder.

As we will see when discussing the Turing completeness of Rule 110, in order for it to comply with our definition of local universality, we have to further generalize the form of the CA's background (quiescent state is too restrictive) and adjust equation (3) by appropriate shifts, since the information in the CA's space-time diagram can, in general, be shifted around by the dynamics.

Local simulation: formal definition

Definition D.4 (Admissible background)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a CA. We say that $B \in S^{\mathbb{Z}^d}$ is a* basic admissible background *if there exists a packing $(\mathcal{P}, V)$ of $\mathbb{Z}^d$ with $|\mathcal{P}| = m$ and a cellular transformaton $G : \{0\}^{\mathbb{Z}^d} \to (S^m)^{\mathbb{Z}^d}$ such that $B = (o^{\langle \mathcal{P}, V \rangle})^{-1} \circ G(0^{\mathbb{Z}^d})$. We say that $B \in S^{\mathbb{Z}^d}$ is an* admissible background *if there exists a finite partition of $\mathbb{Z}^d$ by convex polyhedral regions $R_1, \ldots, R_k$ delimited by linear inequalities with rational coefficients and a sequence of basic backgrounds $B_1, \ldots, B_k$ such that for each $i \in \{1, \ldots, k\}$ it holds that $B\big|_{R_i} = B_i\big|_{R_i}$.*

An example of an admissible background in dimension 2 is illustrated in Fig. 15.

Definition D.5 (Local simulation)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a CA and let $\mathcal{T} : C_{\mathcal{T}} \to C_{\mathcal{T}}$ be a Turing machine. We say that $\mathcal{A}$ simulates $\mathcal{T}$ if there exists a finite collection of admissible backgrounds $\{B_1, \ldots, B_k\}$, a mapping $\mathcal{E} : C_{\mathcal{T}} \to S^{\mathbb{Z}^d}$, a partial mapping $\mathcal{D} : S^{\mathbb{Z}^d} \rightharpoonup C_{\mathcal{T}}$, a partial mapping $\tau : S^{\mathbb{Z}^d} \to \mathbb{N}$, and two vectors $v, w \in \mathbb{Z}^d$ such that:*

$$\mathcal{D} \circ \left(\sigma_w \circ (\sigma_v \circ F)^\tau\right)^n \circ \mathcal{E} = T^n \qquad \text{for every } n \in \mathbb{N}.$$

The mappings $\mathcal{E}$ (encoder), $\mathcal{D}$ (decoder), and τ (delay function) must satisfy the following:

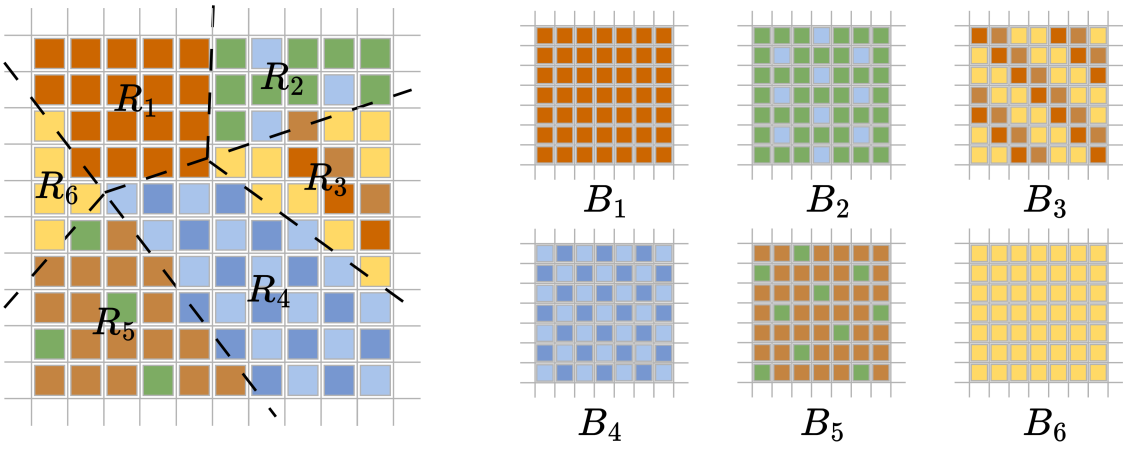


Figure 15: Example of an admissible background in $\mathbb{Z}^2$ delimited by six convex regions (left) and corresponding basic backgrounds (right).

- *The encoder $\mathcal{E} : C_{\mathcal{T}} \to S^{\mathbb{Z}^d}$ is implemented by a Turing machine with $k+2$ tapes. The first tape contains the input configuration $c \in C_{\mathcal{T}}$. The second tape is empty, this is where the output will be written. The tapes with indices $3, \dots, k+2$ each contain the descriptions of admissible backgrounds $\{B_1, \dots, B_k\}$. The Turing machine has one head that can move across the tapes and halts for every input configuration. Upon halting, the output tape will contain an R-pattern over S where $R = [-k_c, k_c]^d$ for some $k_c \in \mathbb{N}$ and further, the output tape contains the description of an index $i \in \{1, \dots, k\}$ of an admissible background. The output yields a configuration $\mathcal{E}(c) \in S^{\mathbb{Z}^d}$ defined by $\mathcal{E}(c)\big|_R = c'$ and $\mathcal{E}(c)\big|_{\mathbb{Z}^d \setminus R} = B_i$.*

- *The decoder $\mathcal{D} : S^{\mathbb{Z}^d} \rightharpoonup C_{\mathcal{T}}$ is implemented by a Turing machine with two tapes – the input and output tape. The input tape contains the configuration $c' \in S^{\mathbb{Z}^d}$ with the head positioned at $0 \in \mathbb{Z}^d$. Upon halting, $\mathcal{D}$ outputs on the second tape a configuration $c \in C_{\mathcal{T}}$. If c' is not in the domain of $\mathcal{D}$, then the Turing machine need not halt.*

- *The delay function τ has the same domain as $\mathcal{D}$. It is also implemented by a Turing machine with two tapes – the input and output tape. Upon halting, the output tape contains the binary representation of the integer $\tau(c') \in \mathbb{N}$.*

Moreover, we require that there exists a computable function $\beta : \mathbb{N} \to \mathbb{N}$ such that $\mathcal{E}(c)$ runs in time $O\big(\beta(|c|)\big)$, and $\mathcal{D}(c')$ and $\tau(c')$ run in time $O\big(\beta(|\mathcal{D}(c)|)\big)$. In such a case, we say that the simulation runs in time $O(\beta(n))$.

We notice that due to the time-complexity bound on the encoder, for the central pattern encoding c, it holds that $k_c \leq \beta(n)$. Similarly, the time-complexity bound on the decoder implies that $\mathcal{D}$ only depends on a finite region of the input c' whose size is also bounded by $\beta(n)$.

Definition D.6 (Local universality)**.** *We say that a CA is* locally universal *if it can locally simulate a universal Turing machine.*

Remark D.7. *We remark that there might appear future constructions of Turing-complete CAs that do not agree with Definition D.5 exactly. For instance, the notion of an admissible background could be relaxed to allow for more general geometrical partitions and patterns. Alternatively, one*

may require a CA $\mathcal{A}$ to only simulate every k-th step of a Turing machine rather than every step exactly. However, we argue that Definition D.5 can readily be generalized to account for such examples without having to undergo a fundamental change.

One important generalization we will use later is that of *fractional shifts*: for a 1D system and $p, q \in \mathbb{N}$, let $\sigma_{\frac{p}{q}}$ denotes a mapping that every q steps shifts a given configuration by p cells to the left. This notion can be generalized to higher dimensions in a straightforward way.

Verifying that Definition D.5 is compatible with every single construction of a Turing-complete CA from the literature is clearly outside of the scope of this paper. In what follows, we focus on one of the most elaborate works: the proof of the Turing completeness of Rule 110 [13] – a CA whose local rule is so compact that in order to show its computational universality, the capacities of the corresponding encoder and decoder have been stretched much further compared to other constructions. Showing directly that Rule 110 can simulate a universal Turing machine would be quite cumbersome and instead, the proof in [13] follows a series of reductions, arriving at a much simpler, yet universal model of computation called a cyclic tag system. Before we proceed with a detailed discussion of Rule 110's Turing completeness, we introduce a series of computational systems and discuss their universality.

D.2 Computational systems and their simulation

We briefly introduce all "intermediate" computational systems used in the proof of Rule 110's universality, and we define the notion of one computational system simulating another.

Definition D.8 (Clockwise Turing machine)**.** *A clockwise Turing machine is like a standard single-tape Turing machine except for the following details:*

(i) the tape is finite and circular;

(ii) the head moves only clockwise on the tape;

(iii) before moving to the next cell, the head can either write a single symbol into the current cell, or split the current cell in two and write two new symbols (thus prolonging the tape).

Analogously to regular Turing machines, for each clockwise Turing machine $\mathcal{T}$ we define its configuration space $C_{\mathcal{T}}$ and associate $\mathcal{T}$ with a dynamical system $(C_{\mathcal{T}}, T)$.

Definition D.9 (Tag system)**.** *A tag system is defined by a finite set of symbols Σ and a transition function $f : \Sigma \to \Sigma^*$. Given a tape configuration $s_1 \cdots s_n$ of symbols from Σ, the tag system updates it as follows:*

(i) the symbols s_1, s_2 are deleted from the start of the tape;

(ii) the word $f(s_1)$ is appended at the end of the tape.

The configuration space of a tag system $\mathcal{T}$ with alphabet Σ is defined as $C_{\mathcal{T}} = \Sigma^*$ and the local transition function f induces the map $T : \Sigma^* \to \Sigma^*$. Again, we can associate with $\mathcal{T}$ the dynamical system $(C_{\mathcal{T}}, T)$.

Definition D.10 (Cyclic tag system)**.** *A cyclic tag system operates on binary configurations (with symbols* Y *and* N*) and is defined by a list of appendants $(\alpha_0, \alpha_1, \ldots, \alpha_{z-1})$ with each $\alpha_i \in \{\mathrm{Y}, \mathrm{N}\}^*$. Given a tape configuration $s_1, \ldots, s_n$ of symbols from $\{\mathrm{Y}, \mathrm{N}\}$, the cyclic tag system at time t updates it as follows:*

(i) s_1 *is removed from the start of the tape*

(ii) if $s_1 = \mathrm{Y}$ *then the word* $\alpha_{t \bmod z}$ *is appended at the end of the tape, otherwise, nothing gets appended.*

After the update, time increases to $t+1$*. Initially, the system starts at time* $t = 0$*. The configuration space of a cyclic tag system* $\mathcal{T}$ *with appendant list* $(\alpha_0, \alpha_1, \ldots, \alpha_{z-1})$ *is defined as* $C_{\mathcal{T}} = \{\mathrm{Y},\mathrm{N}\}^* \times \{0\} \times \{\mathrm{Y},\mathrm{N}\}^*$ *where a configuration* $c = c_1 0 c_2$ *represents the tape content by* c_1 *and current appendant by* c_2*. The update rule induces the map* $T : C_{\mathcal{T}} \to C_{\mathcal{T}}$ *which simply updates the tape and the new appendant. Again, we can associate with* $\mathcal{T}$ *the dynamical system* $(C_{\mathcal{T}}, T)$*.*

Analogously to local simulation, below we define what it means for a computational system to simulate another.

Definition D.11. *Let* $\mathcal{T}_1 = (C_{\mathcal{T}_1}, T_1)$ *and* $\mathcal{T}_2 = (C_{\mathcal{T}_2}, T_2)$ *be computational systems (each can be a Turing machine, a clockwise Turing machine, a tag system or a cyclic tag system). We say that* $\mathcal{T}_2$ *simulates* $\mathcal{T}_1$ *if there exists a mapping* $\mathcal{E} : C_{\mathcal{T}_1} \to C_{\mathcal{T}_2}$*, a partial mapping* $\mathcal{D} : C_{\mathcal{T}_2} \rightharpoonup C_{\mathcal{T}_1}$*, and a partial mapping* $\tau : C_{\mathcal{T}_2} \rightharpoonup \mathbb{N}$ *such that:*

$$\mathcal{D} \circ (T_2^{\tau})^n \circ \mathcal{E} = T_1^n \qquad \text{for every } n \in \mathbb{N}.$$

The mappings $\mathcal{E}$ *(encoder),* $\mathcal{D}$ *(decoder), and* τ *(delay function) must satisfy the following:*

- *The encoder* $\mathcal{E} : C_{\mathcal{T}} \to S^{\mathbb{Z}^d}$ *is implemented by a Turing machine with two tapes. The first tape contains the input configuration* $c \in C_{\mathcal{T}_1}$*. The second tape is initially empty, and upon halting, it contains* $\mathcal{E}(c) \in C_{\mathcal{T}_2}$*.*
- *The decoder* $\mathcal{D} : C_{\mathcal{T}_2} \rightharpoonup C_{\mathcal{T}_1}$ *is implemented by a Turing machine with two tapes – the input and output tape. The input tape contains the configuration* $c' \in S^{\mathbb{Z}^d}$*, if* c' *is in the domain of* $\mathcal{D}$*, the Turing machine halts with the initially empty output tape containing the configuration* $\mathcal{D}(c') \in C_{\mathcal{T}_1}$*. If* c' *is not in the domain of* $\mathcal{D}$*, then the Turing machine need not halt.*
- *The delay function* τ *has the same domain as* $\mathcal{D}$*. It is also implemented by a Turing machine with two tapes – the input and output tape. Upon halting, the output tape contains the binary representation of the integer* $\tau(c') \in \mathbb{N}$*.*

Moreover, we require that there exists a computable function $\beta : \mathbb{N} \to \mathbb{N}$ *such that* $\mathcal{E}(c)$ *runs in time* $O\big(\beta(|c|)\big)$*, and* $\mathcal{D}(c')$ *and* $\tau(c')$ *run in time* $O\big(\beta(|\mathcal{D}(c')|)\big)$*. In such a case, we say that the simulation runs in time* $O(\beta(n))$*.*

Remark D.12. *Notice that in Definition D.5 of local simulation we have not used any particular property of a Turing machine other than the fact that it induces a dynamical system* $(C_{\mathcal{T}}, T)$ *where every configuration* $c \in C_{\mathcal{T}}$ *is a finite sequence of symbols from a finite alphabet. Thus, the definition of local simulation can be readily generalized to cover the cases of a CA locally simulating any computational system defined above.*

Now, we are ready to prove two straightforward, yet very useful results regarding the composition of simulating systems.

Lemma D.13. *Let* $\mathcal{T}_1 = (C_{\mathcal{T}_1}, T_1)$*,* $\mathcal{T}_2 = (C_{\mathcal{T}_2}, T_2)$*, and* $\mathcal{T}_3 = (C_{\mathcal{T}_3}, T_3)$ *be computational systems (each of them can be a Turing machine, a clockwise Turing machine, a tag system or a cyclic tag system). If* $\mathcal{T}_3$ *simulates* $\mathcal{T}_2$ *in time* $O(\beta_2(n))$ *and* $\mathcal{T}_2$ *simulates* $\mathcal{T}_1$ *in time* $O(\beta_1(n))$*, then also* $\mathcal{T}_3$ *simulates* $\mathcal{T}_1$ *in time* $O\left(\max\{\beta_1(n), \beta_2(n)\}\right)$*.*

Proof. Let $\mathcal{E}_1, \mathcal{D}_1, \tau_1$ witness that $\mathcal{T}_2$ simulates $\mathcal{T}_1$ and let $\mathcal{E}_2, \mathcal{D}_2, \tau_2$ witness that $\mathcal{T}_3$ simulates $\mathcal{T}_2$. It is straightforward to verify that $\mathcal{T}_3$ simulates $\mathcal{T}_1$ with the encoder $\tilde{\mathcal{E}} = \mathcal{E}_2 \circ \mathcal{E}_1$, the decoder $\tilde{\mathcal{D}} = \mathcal{D}_1 \circ \mathcal{D}_2$ and with the delay function $\tilde{\tau} = \tau_2 \circ T_1^{\tau_1}$. □

Lemma D.14. *Let $\mathcal{T}_1 = (C_{\mathcal{T}_1}, T_1)$, and $\mathcal{T}_2 = (C_{\mathcal{T}_2}, T_2)$ be computational systems (each of them can be a Turing machine, a clockwise Turing machine, a tag system or a cyclic tag system) and let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a cellular automaton. If $\mathcal{T}_2$ simulates $\mathcal{T}_1$ in time $O(\beta_2(n))$ and $\mathcal{A}$ simulates $\mathcal{T}_2$ in time $O(\beta_1(n))$, then also $\mathcal{A}$ simulates $\mathcal{T}_1$ in time $O\left(\max\{\beta_1(n), \beta_2(n)\}\right)$.*

Proof. Let $\mathcal{E}_1, \mathcal{D}_1, \tau_1$ witness that $\mathcal{T}_2$ simulates $\mathcal{T}_1$ and let $\mathcal{E}_2, \mathcal{D}_2, \tau_2$ and $v, w \in \mathbb{Z}^d$ witness that $\mathcal{A}$ simulates $\mathcal{T}_2$. It is straightforward to verify that $\mathcal{A}$ simulates $\mathcal{T}_1$ with the encoder $\tilde{\mathcal{E}} = \mathcal{E}_2 \circ \mathcal{E}_1$, the decoder $\tilde{\mathcal{D}} = \mathcal{D}_1 \circ \mathcal{D}_2$, with the delay function $\tilde{\tau} = \tau_2 \circ T_1^{\tau_1}$, and with shift vectors v, w. □

D.3 Local universality of Rule 110

We demonstrate that the definition of local CA simulation agrees with previous constructions on one of the most elaborate examples: the Turing completeness of Rule 110. This requires going over the technical details in Cook's proof: whereas he specified the encoder in [15], we will fill in the details about the delay function, decoder, and shifts.

Cook's proof strategy of Rule 110's Turing completeness entails two important results:

1. There exists a universal cyclic tag system (i.e. a cyclic tag system that can simulate a universal Turing machine).

2. Rule 110 can simulate a large class of cyclic tag systems that contains a universal one.

In order to show that Cook's proof complies with our definition of local universality, we have to show that the cyclic tag system can simulate a universal Turing machine according to Definition D.11 and that Rule 110 can simulate the universal tag system according to Definition D.5. Then, using Lemma D.14 we obtain that Rule 110 can locally simulate a universal Turing machine.

D.3.1 Universal Cyclic Tag System

In this section, we briefly discuss the first result. Subsequently, our main focus is the second result, which requires a detailed understanding of a variety of gliders and their collisions within Rule 110's dynamics.

Let $\mathcal{A} \mapsto \mathcal{B}$ denote that system $\mathcal{B}$ can simulate system $\mathcal{A}$. In [13], Cook shows the following series of reductions:

$$\text{Turing machine with alphabet } \{0,1\} \mapsto \text{tag system} \mapsto \text{cyclic tag system}\,.$$

This, however, leads to an exponential slowdown – a problem later solved by Neary and Woods in [14] who showed that a cyclic tag system can actually simulate a Turing machine in polynomial time (a result noted as surprising by Cook [15]). The authors prove the following series of reductions:

$$\text{binary Turing machine} \mapsto \text{binary clockwise Turing machine} \mapsto \text{cyclic tag system}\,.$$

Here, a binary Turing machine is a Turing machine with only two tape symbols. Since a binary Turing machine can simulate an arbitrary Turing machine with at most a constant factor increase in time, it follows that there exists a universal binary Turing machine. We briefly summarize the results of Neary and Woods in the following two lemmas.

Lemma D.15 (Universal clockwise Turing machine [14])**.** *For each Turing machine $\mathcal{T}$ given by (Q, Σ, δ) that runs in time $t(n)$ there exists a clockwise Turing machine $\mathcal{T}'$ that simulates it and runs in time $O(t^2(n))$.*

Proof. We give a proof sketch: Let $\mathcal{T}$ be a Turing machine with arbitrary alphabet Σ and m states. Then, the clockwise Turing machine $\mathcal{T}'$ operates as follows:

1. Right moves of $\mathcal{T}$, when visiting a non-blank symbol, correspond to clockwise moves of $\mathcal{T}'$.
2. When $\mathcal{T}$ moves right and visits a blank symbol, $\mathcal{C}_{\mathcal{T}}$ splits its current cell into two, writing a special placeholder symbol r into the second cell, and proceeds by cycling around the whole tape to end up in r.
3. Left moves of $\mathcal{T}$, when reading a non-blank symbol, correspond to $\mathcal{T}'$ leaving a placeholder symbol l in its current cell and cycling around its whole tape while shifting each symbol one cell clockwise.
4. When $\mathcal{T}$ moves left and visits a blank symbol, $\mathcal{T}'$ splits its current cell into two, writing a special symbol l' into the first cell, and proceeds by cycling around the whole tape until it reaches the symbol l'.

In this case, both the encoder $\mathcal{E}$ and decoder $\mathcal{D}$ are simply the identity mappings. Together with the delay function τ, they run in time $O(n)$. □

Lemma D.16 (Universal cyclic tag system [14])**.** *For each binary clockwise Turing machine $\mathcal{T}$ given by $(Q, \{a, b\}, \delta)$ that runs in time $t(n)$ there exists a cyclic tag system that simulates it and runs in time $O(|Q|t^2(n) \log t(n))$.*

Proof. This is an elaborate proof and we give a rough sketch; for details, see [14]. Let us fix a binary clockwise Turing machine $\mathcal{T}$ with tape symbols $\{a, b\}$ and m states. Let $z = 30m + 61$. We will construct a cyclic tag system $\mathcal{T}'$ with appendants $(\alpha_0, \alpha_1, \ldots, \alpha_{2z-1})$; each $\alpha_i \in \{0, 1\}^*$. Given an initial tape configuration of $\mathcal{T}'$ consisting of the current state q_i, read symbol σ_1 and tape contents $\sigma_1 \cdots \sigma_s \in \{a, b\}^s$ we encode this as a configuration of $\mathcal{B}$ as follows:

$$\underline{\alpha_0}, \alpha_1, \ldots, \alpha_{2z-1} \qquad \langle 1, q_i\rangle\langle\sigma_1\rangle \cdots \langle\sigma_s\rangle\mu^{s'}.$$

The left part shows the appendant list with the current index underlined. The right part shows the cyclic tag system's configuration with $\mu = 10^{z-1}$, $s' = 2^{\lceil \log_2(s)\rceil}$ and $\langle 1, q_i\rangle, \langle\sigma_1\rangle, \ldots, \langle\sigma_s\rangle \in \{0, 1\}^*$.

The goal is to "separate" the encoded read tape symbol $\langle\sigma_1\rangle$ from the rest of the symbols such that $\langle\sigma_1\rangle$ is the only symbol that "knows it has to update itself into a new tape symbol and new state" and it also "knows about the current state q_i". This separation will be achieved by two stages of the simulation process and will make use of the μ symbols as counters. After the separation is finished, the system will arrive into the third stage:

$$\alpha_0, \ldots, \underline{\alpha_{z+10}}, \ldots, \alpha_{2z-1} \qquad \langle 3, q_i\rangle\langle\sigma_1\rangle\langle\not\sigma_2\rangle \cdots \langle\not\sigma_s\rangle\not\mu^{s'}$$

Here, the marked symbols are all binary sequences of length $2z$ and represent tape data and counter symbols that "know they should not update themselves in any way". The only uncrossed symbol is σ_1 which will make the update. But how does $\langle\sigma_1\rangle$ know that $\mathcal{T}'$ is currently in state q_i? Crucially, this is ensured by the encoding $\langle 3, q_i\rangle$: its length is $z + i + 20$ so after $z + i + 20$ steps, the cyclic tag system ends up in state

$$\alpha_0, \ldots, \underline{\alpha_{30i+30}}, \ldots, \alpha_{2z-1} \qquad \langle\sigma_1\rangle\langle\not\sigma_2\rangle \cdots \langle\not\sigma_s\rangle\not\mu^{s'}a$$

where $a \in \{0,1\}^*$ marks the new appendant (its precise form is not important at this level of detail). The main idea is that the update rule of the cyclic tag system now leverages two pieces of information: the current read symbol is encoded in $\langle \sigma_1 \rangle$ and the current state of the simulated Turing machine is encoded in the index of the appendant: $30i + 30$. Suppose for simplicity that the Turing machine, as it moves clockwise, writes only one symbol $\sigma_{s+1} \in \{a, b\}$ and changes to state q_k. Then, the cyclic tag system mimics this transition by reading all $2z$ symbols of $\langle \sigma_1 \rangle$ and adding the appendant $\langle \sigma_{s+1} \rangle \langle 1, q_k \rangle$. To finish, stage 3 cycles through the remaining symbols while "unmarking them". Lastly, the length of appendant a ensures that after a was read, the index of the tag system returns to 0. Thus, at the end of stage 3, the new configuration of $\mathcal{B}$ will be:

$$\underline{\alpha_0}, \alpha_1, \ldots, \alpha_{2z-1} \qquad \langle 1, q_i \rangle \langle \sigma_2 \rangle \cdots \langle \sigma_s \rangle \mu^{s'} \langle \sigma_{s+1} \rangle \,.$$

Apparently, the next read symbol is σ_2. The only difference with the initial encoding is that now, the encoded symbol $\langle \sigma_{s+1} \rangle$ is placed after the counter symbols μ. It is apparent from the details of the simulation that the relative position of encoded tape symbols and counter symbols does not matter, as long as the order of the encoded tape symbols is correct. This sums up the main idea of the construction. If there are currently n symbols on the Turing machine's tape, simulating one stage of the cyclic tag will take $O(n)$ time, and the stages are executed $O(\log n)$ times. Thus, to simulate one step of the clockwise Turing machine, the cyclic tag needs $O(n \log n)$ steps.

We briefly discuss the time and space complexities of the encoder, decoder and delay function. Each encoded object $\langle * \rangle$ has length at most $3z$ where z only depends on the number of the Turing machine's states. Thus, the encoder and decoder time-complexities are in $O(n)$ (with respect to the current length n of the simulated Turing machine's tape), only needing constant space. The time-delay function simply needs to iterate the cyclic tag system until the end of stage 3, outputting the number of time-steps needed to reach a decodable cyclic tag configuration, thus running in time $O(n \log n)$ (with respect to the current length n of the simulated Turing machine's tape) using at most $O(n)$ space. □

Remark D.17. *From the detailed proof in the paper of Neary and Woods [14], it is clear that the cyclic tag system constructed as above satisfies a property that will later become important: when simulating the Turing machine, after every full cycle through the appendant list, at least one non-empty appendant gets appended.*

D.3.2 Rule 110 Simulating Cyclic Tag Systems

In what follows, we discuss Cook's second, much more elaborate result: Rule 110 can simulate a large class of cyclic tag systems that contains a universal one and moreover, Rule 110 only needs a polynomial time delay to do so. Our aim is to show that this result complies with Definition D.5 of local simulation.

Definition D.18 (Admissible tag systems)**.** *We say that a cyclic tag $\mathcal{T}$ is* admissible *if the first appendant is nonempty and every nonempty appendant's length is a multiple of 6. We say that a configuration $c \in C_{\mathcal{T}}$ is* admissible*, if while iterating $\mathcal{T}$ on c, we have a guarantee that each pass through the whole appendant list, at least one non-empty appendant gets appended.*

From Cook's construction, it follows that Rule 110 can simulate only admissible cyclic tag systems from admissible initial configurations. Crucially, the results of Cook, and Neary and Woods, give the following lemma.

Lemma D.19. *There exists a universal cyclic tag system $\mathcal{U}$ that can simulate a universal Turing machine $\mathcal{T}_{\text{univ}}$ with a polynomial time delay that satisfies the following: Let $\mathcal{E}$ be the encoder witnessing the simulation. Then, for every configuration $c \in C_{\mathcal{T}_{\text{univ}}}$, $\mathcal{E}(c)$ and $\mathcal{U}$ are admissible.*

Proof. In the introduction of [15], Cook describes a simple way of converting an arbitrary tag system into one that can simulate it and whose appendant lengths are multiples of 6: we expand each appendant by adding the symbol N five times after every symbol in the appendant, and we expand the tape similarly. Lastly, we expand the list of appendants by adding the empty appendant five times after every original appendant. We illustrate it on an example. A tag system with initial tape containing Y and appendants (Y, N) can be transformed into a tape YNNNNN with appendants $(\text{YNNNNN}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \text{NNNNNN}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$; it is easy to check that the computational dynamics of the two systems will be equivalent. Moreover, it is easy to see that if the configuration c was admissible for the original cyclic tag, then the expanded configuration c' is admissible for the expanded tag.

Let $\mathcal{U}$ be the universal tag system constructed by Neary and Woods which simulates a universal Turing machine via an encoder $\mathcal{E}$. From the details in the work of Neary and Woods (Lemma D.16), it is easy to see that $\mathcal{E}(c)$ is admissible for $\mathcal{U}$. However, it is not the case that every appendant's length is a multiple of 6. It suffices to use the above transformation, to obtain a universal cyclic tag $\mathcal{U}'$ which is admissible, and for which every encoded tape of the universal Turing machine it simulates is also admissible. Moreover, $\mathcal{U}'$ simulates the universal Turing machine in a polynomial time, with encoder, decoder and delay function complexities being polynomial in time and linear in space. □

The general idea of Cook's construction is illustrated in Fig. 16. In the diagram, each row represents a configuration of Rule 110 with time progressing downwards; and each object (tape symbols, ossifiers, appendants) is represented by a group of gliders in Rule 110.

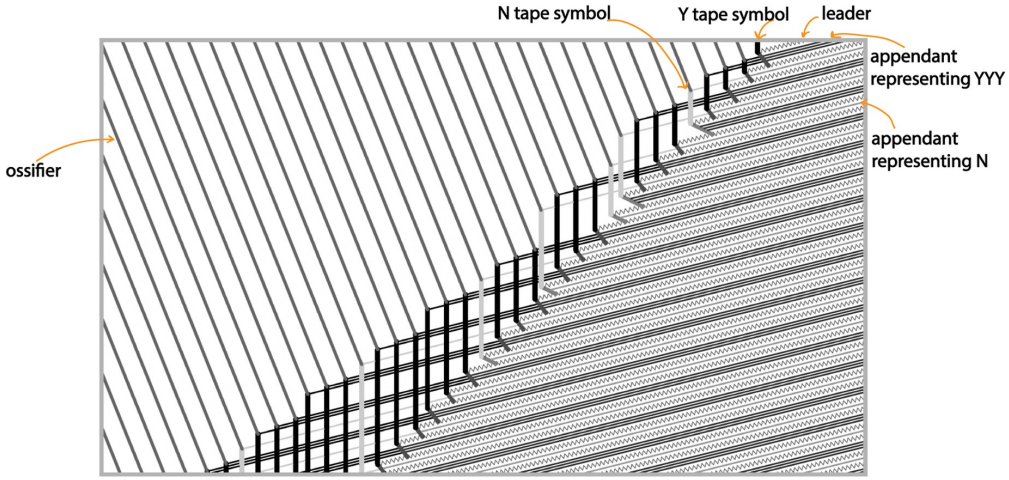


Figure 16: General Scheme of Rule 110 simulating a cyclic tag with appendant list {YYY, N} and with initial tape symbol Y. (Such a cyclic tag is non-admissible and this scheme is only illustrative). Figure is adapted from [13].

- The tape data is encoded in a finite region, and the symbols Y and N represented by static gliders. We note the tape is represented in a "reversed order", with the first symbol on the cyclic tag's tape being the rightmost one.

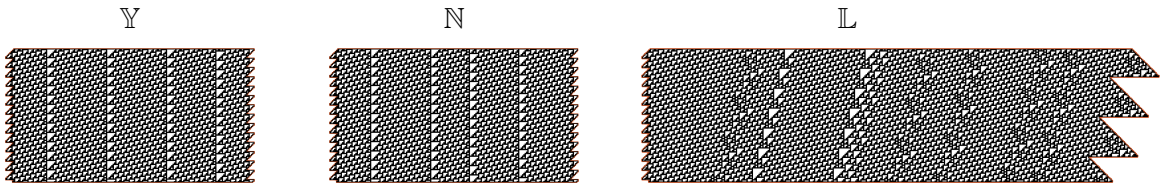
- To the right of the tape data, there is an infinitely repeating periodic sequence which represents the ever repeating cyclic tag's appendant list. Each appendant is represented by a block of gliders moving to the left, each block starts with a "leader" glider, and is followed by gliders representing the actual data.
- Once a leader hits a tape symbol, the symbol is erased by the collision. If it was an N, then the leader gets modified and wipes out all the following appendant data, upon being annihilated by the next leader. If it was a Y, then the leader transforms the gliders into "moving data", which pass through the whole tape region, leaving it intact.
- To the left of the tape data, there is an infinitely repeating periodic sequence of "ossifiers". An ossifier is a glider moving towards the right. Its purpose is to collide with moving data gliders: this collision annihilates both the ossifier and moving data glider, and it creates a corresponding static data symbol.

We note that if the ossifier collides with a tape symbol before hitting a moving data glider, both get annihilated and the construction will not work. Thus, we need a guarantee that if the ossifiers are spaced far enough apart, they will not collide with the tape data. This is exactly where the second condition of a cyclic tag's admissibility comes in.

The gliders representing the different objects of the construction are represented in Figures 17, 18, and 19. One can see that the gliders have various temporal periods. Representing them as "puzzle pieces" ensures that the gliders are connected exactly at such a phase which makes them compatible. This elegant way of representing the building blocks of the encoder was introduced in Cook's paper [15].

D.3.3 Encoder

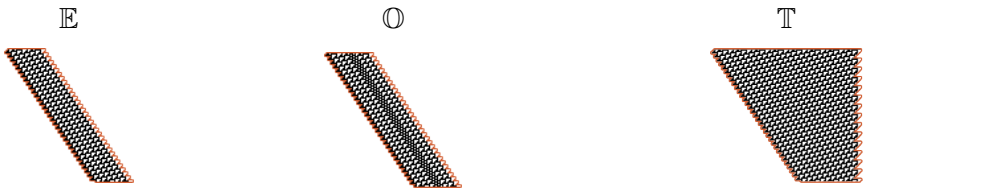
We first describe the construction of the encoder, closely following Cook's approach [15] with slight modifications for soundness and more clarity. The encoded tape consists of three parts: the central part representing the tape symbols, the left part containing a periodic sequence of ossifiers, and the right part containing the periodically repeated representation of the cyclic tag system.



(a) Y tape symbol (b) N tape symbol (c) prepared leader

Figure 17: Central Part of the Tape

Example D.20 (Central Part of the Tape)**.** *Encoder transforms the tape symbols YNN into a finite configuration concatenating the gliders $\mathbb{NNYL}$ (the order of symbols in the encoded configuration is reversed). The rightmost glider is a leader which will be followed by an infinitely repeated pattern encoding the cyclic tag system's appendant list. On the left, the configuration will be prolonged by a repeated pattern of ossifiers.*



(a) ether (b) ossifier (c) ether to tape symbols transition

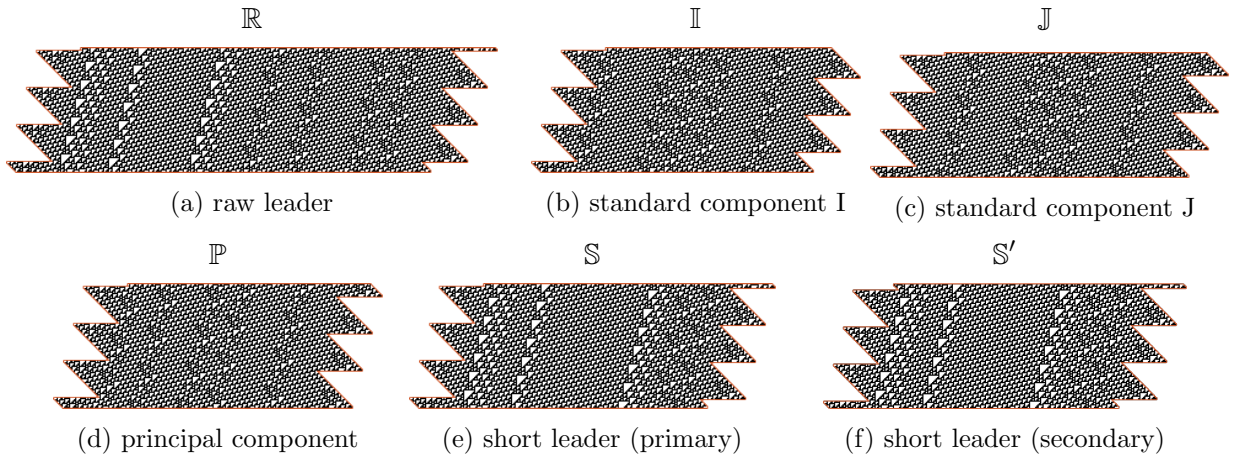
Figure 18: Left Part of the Tape

Left part of the tape The ether $\mathbb{E}$ forms a background pattern through which all gliders move. The sequence $\mathcal{L} = [\mathbb{E}]^{\nu}\mathbb{O}[\mathbb{E}]^{13}\mathbb{O}[\mathbb{E}]^{11}\mathbb{O}[\mathbb{E}]^{12}\mathbb{O}$ encodes four ossifiers; upon colliding with four moving data gliders, they will produce a static tape symbol. The constant ν ensures that the groups of ossifiers are spaced far enough from each other to never collide with tape data by mistake. Explicitly, Cook [15] computes ν as follows:

$$\begin{aligned} v = \quad & 76 \cdot (\text{the total number of } Y\text{s in all appendants}) \\ & + 80 \cdot (\text{the total number of } N\text{s in all appendants}) \\ & + 60 \cdot (\text{the number of nonempty appendants}) \\ & + 43 \cdot (\text{the number of empty appendants}) \end{aligned}$$

The transition pattern $\mathbb{T}$ simply connects the static tape data to the ossifiers. Lastly, we have to make sure the first group of ossifiers is spaced far enough not to hit the tape data by mistake; this is accounted for by the constant ν' (a conservative estimate is to make it larger than $\nu + 14 \cdot$ (the total number of tape symbols)).

Example D.21 (Left Part of the Tape)**.** *The left part of the tape, together with the tape symbols YNN is encoded as $\cdots\mathcal{L}\mathcal{L}[\mathbb{E}]^{\nu'}\mathbb{T}\mathbb{N}\mathbb{N}\mathbb{Y}\mathbb{L}$ where $\mathcal{L} = [\mathbb{E}]^{\nu}\mathbb{O}[\mathbb{E}]^{13}\mathbb{O}[\mathbb{E}]^{11}\mathbb{O}[\mathbb{E}]^{12}\mathbb{O}$. We can assume that the last symbol of $\mathbb{L}$ is exactly at the position 0.*



(a) raw leader (b) standard component I (c) standard component J

(d) principal component (e) short leader (primary) (f) short leader (secondary)

Figure 19: Right Part of the Tape

Right part of the tape The gliders in Fig. 19 are used to encode the cyclic tag system's appendant list using the following rules. Each Y symbol is represented by the gliders $\mathbb{III}$, each N by

$\mathbb{IJ}$. In the first symbol of each appendant, the first standard component $\mathbb{I}$ is replaced by primary component $\mathbb{P}$. There is a raw leader (either long or short) between every two appendants. Let a_i, a_j be two consecutive appendants; we use raw (long) leader $\mathbb{R}$ if $a_j \neq \emptyset$, we use $\mathbb{S}$ if $a_i \neq \emptyset$ and $a_j = \emptyset$ and we use $\mathbb{S}'$ if $a_i = \emptyset$ and $a_j = \emptyset$ [1].

Example D.22 (Right Part of the Tape)**.** *A cyclic tag system with appendant list given by* $\{\text{YNN}, \text{NYYN}, \emptyset, \emptyset, \emptyset\}$ *is encoded as:* $\mathcal{R} = \underbrace{\mathbb{PIIJIJ}}_{\text{YNN}}\,\mathbb{R}\,\underbrace{\mathbb{PJIIIIIJ}}_{\text{NYYN}}\,\mathbb{SS}'\mathbb{S}'\mathbb{R}$.

Lemma D.23. *There exists an encoder $\mathcal{E}$ satisfying the constraints of local simulation definition, which transforms the tape of an admissible cyclic tag into an initial configuration of Rule 110 that agrees with Cook's construction. Moreover, the encoder's complexity is linear in time and constant in space.*

Proof. Let $\mathcal{T}$ be an admissible cyclic tag system and let $C \subseteq C_{\mathcal{T}}$ be a set of admissible configurations. We construct an encoder $\mathcal{E} : C \to \mathbf{2}^{\mathbb{Z}}$ which satisfies the constraints of local simulation Definition D.5 as follows.

Intuitively, the role of the encoder is clear: the Turing machine that implements it transforms the input $s_1 \cdots s_n \in C$ into the central part of Rule 110 configuration using the gliders in Fig. 17. Then, this finite sequence is embedded into an admissible background consisting of two regions: to the left we have a periodic repetition of the sequence $\mathcal{L}$ of ossifiers, to the right there is an ever repeating sequence $\mathbb{R}$ representing $\mathcal{T}$'s appendant list. This concludes the idea of the encoder.

Each glider is constant in size, and we only need a constant number of gliders to represent each symbol of the cyclic tag. For each glider, the encoder has to choose a compatible phase, which takes constant amount of time. Thus, the encoder works in time $O(n)$ and in constant space. □

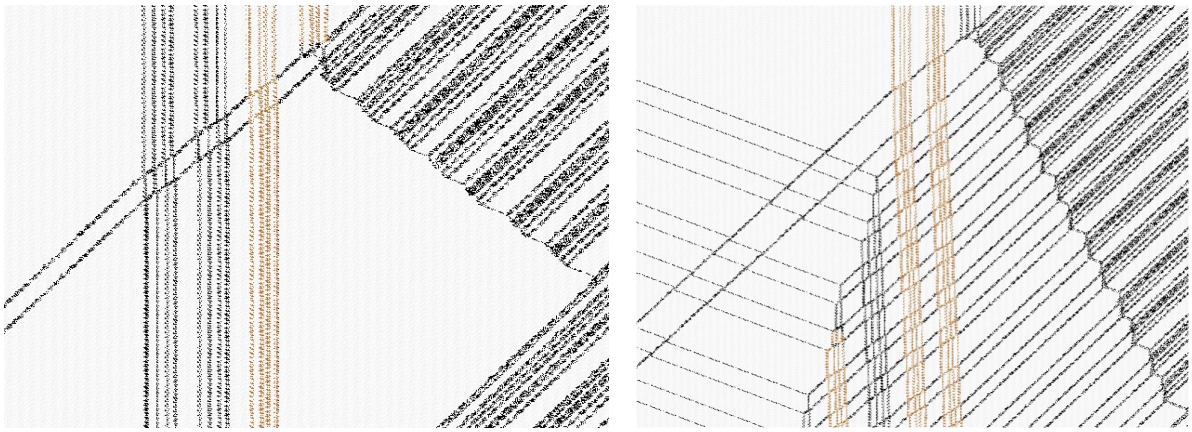
D.3.4 Delay Function and Decoder

Given an admissible cyclic tag system $\mathcal{T}$ and an admissible initial configuration $c \in C_{\mathcal{T}}$, Cook's outline of the proof in [16] guarantees that once we iterate Rule 110 on the encoded configuration $\mathcal{E}(c)$, the collisions of the Rule 110's gliders will mimic the dynamics of $\mathcal{T}$, as presented in Fig. 16. This becomes intuitively clear when observing the dynamics of Rule 110 from an encoded configuration. We illustrate the main types of glider interactions in Fig. 20. To build a deeper intuition for the dynamics, we suggest that the reader explore the interactive library we implemented in order to visualize Rule 110 simulating arbitrary admissible tag systems available here.

In Cook's construction, it is not specified what are the times at which to read out the current cyclic tag system's tape symbols nor what is the particular form of the decoder. We describe both in more detail below.

Delay function τ The main idea behind the algorithm computing τ is simple: given a configuration c which represents the current tape content and a current appendant about to hit the current first tape symbol, τ should ensure that once we iterate Rule 110 $\tau(c)$ times from c, the current appendant will have interacted with the first tape symbol, resulting in either a complete transformation of the appendant into moving data (if the symbol was Y) or a complete deletion of the appendant (if the symbol was N). This ensures that after $\tau(c)$ iterations, all symbols from the update tape are present in the Rule 110's configuration as either static tape data or moving

[1] We highlight a difference with Cook's construction: he only introduces the short leader $\mathbb{S}$. However, from our experiments, it appears this does not suffice since placing two components $\mathbb{SS}$ next to each other representing two empty appendants does not give the correct dynamics of the simulation.



(a) Tape contains NNYY; as the leader hits the symbol N, it emits a "rejector" that erases the following appendant and then gets absorbed by the next raw leader. The collision of the leader with a tape symbol produces a pair of "garbage gliders" that propagate to the left and pass through all ossifiers without interfering with them.

(b) Tape contains YNN; as the leader hits the first Y, it emits an "acceptor" that transforms the following appendant into moving data. The moving data pass through the static tape data, leaving it intact. Eventually, they collide with ossifiers, each four-tuple transformed into a static tape symbol. Again, a pair of "garbage gliders" is produced.

Figure 20: Main types of glider interactions in Rule 110's simulation of a cyclic tag system. Both figures were generated by the PyCA Rule 110 library we implemented here. For better visibility, the ether background is abstracted away, static tape gliders representing Y are colored black, gliders representing N are colored orange.

data. Then, the decoder simply maps the static and moving tape data into the corresponding tape symbols. We portray a schematic illustration of τ in Fig. 21.

We present a very simplified outline of τ's implementation below.

Algorithm 1: Algorithm for the delay function.

Data: configuration $c \in \{0,1\}^{\mathbb{Z}}$ where the distance between the current leftmost leader l_t and the rightmost tape symbol is less than 300.

Result: $\tau \in \mathbb{N}$ the number of iterations such that $c' = F^{\tau}(c)$ is the configuration where the distance between the current leftmost leader l_{t+1} and the rightmost tape symbol is less than 300.

```
τ = 0;
while dist(next tape symbol, leader l_{t+1}) > 300 do
    τ = τ + 1;
    c = F(c);
return τ;
```

Decoder The algorithm implementing the decoder is straightforward: $\mathcal{D}$ receives a configuration where all current tape symbols are present as either static data or moving data. The decoder simply has to first sweep the tape right to left and decode all static data into either Y or no symbols, and afterwards, it has to sweep the tape from left to right, decoding all moving data gliders. A particularly simple implementation is suggested below:

Lemma D.24. *Let $c \in C_{\mathcal{T}}$ be a configuration encoding n symbols of thy cyclic tag's tape, to the right of which is a leader l_1 about to hit the first tape symbol. Let i be the index delimiting the start of l. Then, both τ and the decoder only need access to $c_{[i-k,i+k]}$ where $k \in O(n)$.*

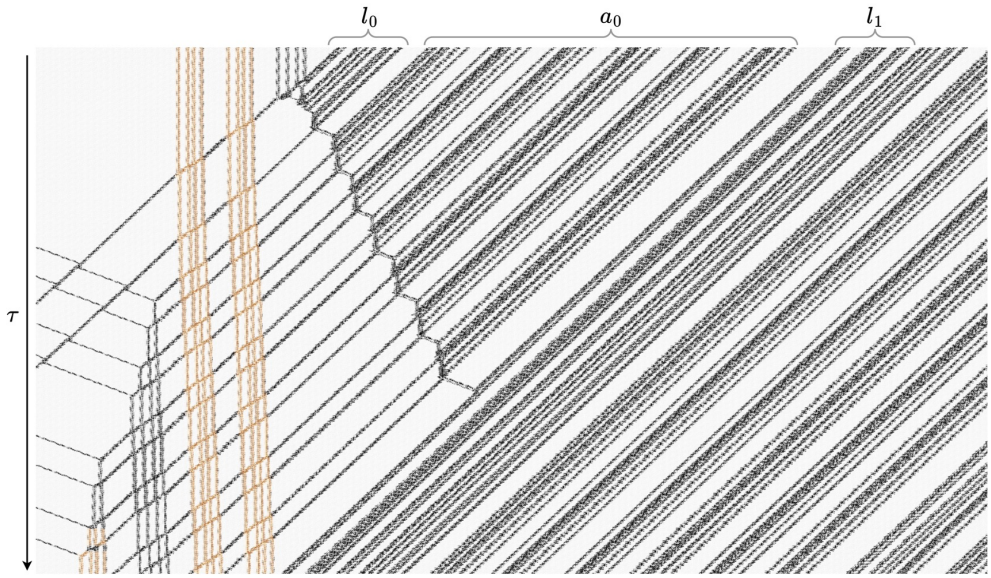


Figure 21: Space-time diagram of Rule 110 illustrating the delay function τ. The first row shows a finite region of a configuration $c \in \mathbf{2}^{\mathbb{Z}}$ where the leftmost leader l_0 is about to hit the rightmost tape symbol Y. The last row shows a portion of the configuration $F^{\tau}(c)$, where the next leader l_1 is about to hit the next tape symbol N. We have guarantee that in $F^{\tau}(c)$ the whole appendant a_0 has been processed and was either deleted, or transformed into moving data that are located to the left of the current rightmost tape symbol.

Algorithm 2: Algorithm for the decoder $\mathcal{D}$.

Data: configuration $c \in \{0,1\}^{\mathbb{Z}}$ where the distance between the current leftmost leader l_t and the rightmost tape symbol is less than 300.
Result: The corresponding tape content $s_1 \dots s_n \in C_{\mathcal{T}}$ of the tag system $\mathcal{T}$.

find $i \in \mathbb{N}$ the position of the leader l_t;
find $j \in \mathbb{N}$ the position of the rightmost ossifier in c;
crop the configuration $c' = c[j, i]$;
while *there are moving data present in* c' **do**
    if c' *contains no ossifiers* **then**
        $c' = \mathcal{L}c'$
    $c' = F(c')$;
reading c' from right to left, match every pattern $\mathbb{Y}$ and $\mathbb{N}$;
return the corresponding symbols

Proof. To the left of i, both τ and the decoder need access to all the static and moving tape data. Let $\nu \in \mathbb{N}$ be the constant number of ether blocks between two consecutive ossifier groups. This guarantees that two consecutive static data symbols are no more than ν ether blocks apart. Thus, the distance between two static tape symbols is bounded by $K_0 = \nu \cdot |\mathbb{E}|$. And the distance from i to the first ossifier is bounded by $K_0 \cdot (n+2)$. To the right, τ and the decoder need access to the next leader l_{t+1}. The distance from l_t to l_{t+1} is bounded by the total length of the encoded cyclic tag pattern, which is some constant K_1.

Therefore, both τ and the decoder need access to the finite part of the configuration delimited to the left by an ossifier and to the right by the leader l_{t+1}: $c_{[i-K_0\cdot(n+2), i+K_1]}$. □

The last and most cumbersome detail to discuss is the following: We have already argued that both the delay function and decoder only need to read a finite portion of the configuration whose size linearly depends on n. This is clear if we center the read-out window around the leftmost's leader position. However, in the definition of local simulation, the readout window is centered around 0. Thus, we have to ensure there exists appropriate shift vectors v, w such that whenever we look at the first leader in the configuration $\sigma_w \circ (\sigma_v \circ F)^\tau$, its distance from 0 is bounded by some constant (independent of the total number of iterations).

We discuss that this is possible in detail below. The leader, as well as all gliders in the right periodic part of the encoded configuration, moves at a constant speed of $v = \frac{4}{15}$; i.e. every 15 time-steps it moves 4 cells to the left. Let $K_a = |\mathcal{R}|$ be the length of the finite sequence encoding the cyclic tag's appendant list and let z be the number of appendants in the list. We denote by $w = \frac{K_a}{z}$ the average appendant length.

Lemma D.25. *Let $\mathcal{T}$ be an admissible tag system and let $c \in C_\mathcal{T}$ be an admissible initial configuration. Let $c_0 = \mathcal{E}(c)$ be the encoded configuration of Rule 110 where $\mathcal{E}$ is the encoder described above. Let $K_a = |\mathcal{R}|$ be the length of the finite sequence encoding the cyclic tag's appendant list and let z be the number of appendants in the list. We let $w = \frac{K_a}{z}$ be the average appendant length and set $v = -\frac{4}{15}$. Then, for each iteration $t \in \mathbb{N}$, the configuration $c_t = \sigma_w \circ (\sigma_v \circ F)^\tau \circ \mathcal{E}(c)$ satisfies the following:*

1. *the leftmost leader l_t in c_t is "about to hit" the rightmost static tape symbol; i.e., their distance is less than 300*
2. *the index i_t delimiting the start of l_t satisfies $|i_t| \leq 2K_a$.*

The lemma implies that if the decoder or delay function receive the configuration c_t, in order to decode or compute the time delay, they only require access to a finite window of the configuration centered around 0, whose size is bounded by the current number of tape symbols.

Proof. Both points 1 and 2 clearly hold for c_0 from the definition of the encoder with $i_0 = 0$. From the definition of τ, it is clear that c_1 encodes the situation when the next leader l_1 as about to hit the current rightmost tape symbol, so clearly 1. is satisfied. In the configuration $F^\tau(c_0)$ the leader l_2 travelled with the speed of 4 cells to the left every 15 time-steps. Thus, shifting the configuration by $(\sigma_v \circ F)^\tau(c_0)$ ensures that the position of l_1 in c_0 and c_1 is identical. If the distance between l_0 and l_1 was exactly $w = \frac{K_a}{z}$ then the final shift $\sigma_w \circ (\sigma_v \circ F)^\tau(c_0)$ would ensure that the position of l_1 in c_1 is again exactly 0. This might not be the case in general, resulting in the more general bound $|i_1| \leq 2K_a$. However, after iterating through all z appendants, we have a guarantee that $i_z = 0$. Thus, every z iterations of the simulation, the current leader's position returns to 0. This finishes the outline of the proof. □

For our exact implementation of the encoder, τ, and decoder in Python, see rule110decoder.

Lemma D.26. *The encoder, decoder, and time-delay witnessing that Rule 110 simulates a universal cyclic tag system have polynomial time complexity, and linear space complexity.*

Proof. The encoder's complexity was discussed above. Suppose the configuration c encodes n tape symbols of the cyclic tag and is ready to be decoded.

The decoder, starting at index 0, requires $O(n)$ time to find the leftmost leader and rightmost ossifier, delimiting a finite portion of the configuration $c_{[i,j]}$ it needs for its computation whose size is in $O(n)$. Next, the decoder as to transform all moving data into static data. This happens in $O(n)$ iterations of Rule 110. Each iteration takes time $O(n)$ to simulate. Finally, the decoder has

to sweep the whole configuration, matching gliders to decode into tape symbols. This takes $O(n)$ time-steps. Thus, the overall time-complexity of the decoder is $O(n^2)$. The decoder only needs to store the current configuration throughout its simulation of Rule 110, resulting in the space complexity of $O(n)$.

Finally, the time-delay function, starting at index 0, also requires $O(n)$ time to find the second leftmost leader and rightmost ossifier, delimiting a finite portion of the configuration $c_{[i,j]}$ it needs for its computation whose size is in O(n). Then, the delay function iterates Rule 110 on this finite region until this leader is about to hit the next static tape symbol, this happens in $O(n)$ iterations. Therefore, the overall time-complexity of the time delay is $O(n^2)$, and the space complexity is $O(n)$. □

We can summarize the previous discussion in the following result.

Theorem D.27. *Rule 110 is locally universal. The time required to simulate one step of a universal Turing machine is polynomial. The encoder, decoder, and time-delay functions witnessing this relation require polynomial time and linear space.*

Proof. From the results of Cook [15] and Woods and Neary [14] we have a guarantee that there exists an admissible tag system $\mathcal{U}$ which simulates a universal Turing machine using a subset of its configurations $C \subseteq C_{\mathcal{U}}$ which are all admissible.

Further, Cook's results guarantee that any $c \in C$ can be encoded into Rule 110's configuration such that Rule 110 iterated on $\mathcal{E}(c)$ mimics the dynamics of $\mathcal{U}$ according to the diagram in Fig. 16. From our detailed discussion above, it is easy to verify that $\mathcal{E}$ satisfies the encoder conditions in the definition of local simulation.

We specify the delay function and decoder by pseudocodes 1 and 2, showing both functions can be implemented by a Turing machine. Lemma D.24 shows that both τ and $\mathcal{D}$ only need access to a finite part of their input configuration whose size is proportional to the current number of tape symbols. Further, Lemma D.25 shows that the readout window can be centered around 0 by choosing appropriate shift vectors for the simulation.

From Lemma D.26 and previous discussions, we know that along the series of reductions shown in Fig. 22, at every step, the encoder, decoder and time-delay have polynomial time-complexity and linear space complexity. Therefore, this also holds for their composition. □

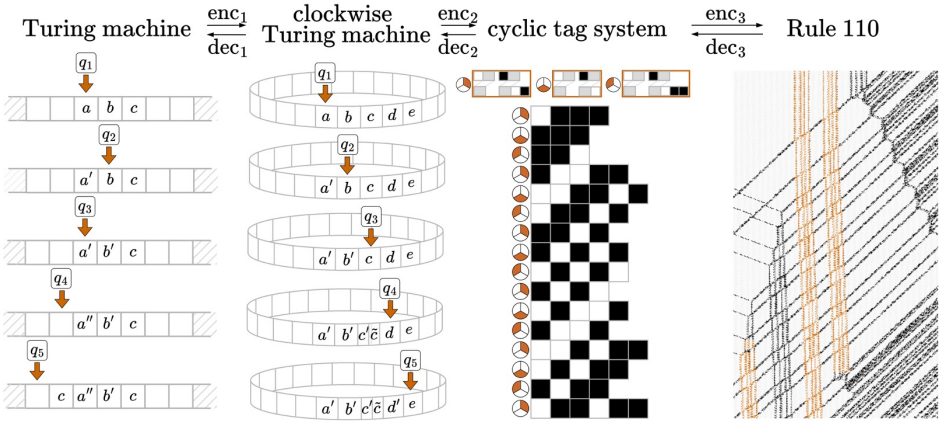


Figure 22: A Scheme illustrating the series of reductions all the way from a universal Turing machine to Rule 110.

D.4 Local vs. global Universality

The goal of this section is to show that global universality is a strictly stronger condition than local universality. First, we show that each globally universal CA is also locally universal. Second, we highlight an example of a 1D reversible CA which is locally universal but not globally universal.

Lemma D.28. *Let $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ and $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be cellular automata and $\mathcal{T} = (C_{\mathcal{T}}, T)$ a computational system (a Turing machine, cyclic tag, etc.). Assume that $\mathcal{B}$ globally simulates $\mathcal{A}$ and $\mathcal{A}$ locally simulates $\mathcal{T}$. Then, $\mathcal{B}$ locally simulates $\mathcal{T}$.*

Proof. Let $\mathcal{E}_1 : C_{\mathcal{T}} \to S^{\mathbb{Z}^d}$ be the encoder and $\mathcal{D}_1 : S^{\mathbb{Z}^d} \rightharpoonup C_{\mathcal{T}}$ the decoder that witness $\mathcal{A}$ locally simulating $\mathcal{T}$ with time-delay function $\tau : S^{\mathbb{Z}^d} \rightharpoonup \mathbb{N}$ and shift vectors $v, w \in \mathbb{Z}^d$; we have:

$$\mathcal{D}_1 \circ \left(\sigma_w \circ (\sigma_v \circ F)^{\tau}\right)^n \circ \mathcal{E}_1(s) = T^n(s) \qquad \text{for all } n \in \mathbb{N} \text{ and all } s \in C_{\mathcal{T}}. \tag{4}$$

Let $\mathcal{E}_2 : S^{\mathbb{Z}^d} \to T^{\mathbb{Z}^d}$ be the encoder and $\mathcal{D}_2 : T^{\mathbb{Z}^d} \rightharpoonup S^{\mathbb{Z}^d}$ the decoder witnessing that $\mathcal{B}$ globally simulates $\mathcal{A}$ with delay constant $t \in \mathbb{N}$ and shift vector $u \in \mathbb{Z}^d$; we have:

$$\mathcal{D}_2 \circ \sigma_u \circ G^t \circ \mathcal{D}_2^{-1}(c) = F(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}. \tag{5}$$

Then, (5) implies:

$$\mathcal{D}_2 \circ \sigma_{M\cdot v+u} \circ G^t \circ \mathcal{D}_2^{-1}(c) = \sigma_v \circ F(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

And further, due to Lemma C.6:

$$\mathcal{D}_2 \circ \left(\sigma_{M\cdot v+u} \circ G^t\right)^{\tau(c)} \circ \mathcal{D}_2^{-1}(c) = \sigma_v \circ F^{\tau(c)}(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

We define a new delay-function $\tau' : T^{\mathbb{Z}^d} \rightharpoonup \mathbb{N}$ as $\tau'(c') = \tau(\mathcal{D}_2(c'))$ for $c' \in T^{\mathbb{Z}^d}$. Then

$$\mathcal{D}_2 \circ \left(\sigma_{M\cdot v+u} \circ G^t\right)^{\tau'} \circ \mathcal{D}_2^{-1}(c) = (\sigma_v \circ F)^{\tau}(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d},$$

and

$$\mathcal{D}_2 \circ \sigma_{M\cdot w} \circ (\sigma_{M\cdot v+u} \circ G^t)^{\tau'} \circ \mathcal{D}_2^{-1}(c) = \sigma_w \circ (\sigma_v \circ F)^{\tau}(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

Again, iterating the argument from Lemma C.6 we have that for all $n \in \mathbb{N}$:

$$\mathcal{D}_2 \circ \left(\sigma_{M\cdot w} \circ (\sigma_{M\cdot v+u} \circ G^t)^{\tau'}\right)^n \circ \mathcal{D}_2^{-1}(c) = \left(\sigma_w \circ (\sigma_v \circ F)^{\tau}\right)^n(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

And thus, as a special case, it also holds for all $n \in \mathbb{N}$ that:

$$\mathcal{D}_2 \circ \left(\sigma_{M\cdot w} \circ (\sigma_{M\cdot v+u} \circ G^t)^{\tau'}\right)^n \circ \mathcal{E}_2(c) = \left(\sigma_w \circ (\sigma_v \circ F)^{\tau}\right)^n(c) \qquad \text{for all } c \in S^{\mathbb{Z}^d}.$$

Substituting the last expression into (4) we get:

$$\mathcal{D}' \circ \left(\sigma_{M\cdot w} \circ (\sigma_{M\cdot v+u} \circ G^t)^{\tau'}\right)^n \circ \mathcal{E}'(s) = T^n(s) \qquad \text{for all } s \in C_{\mathcal{T}},$$

where $\mathcal{E}' = \mathcal{E}_2 \circ \mathcal{E}_1$ and $\mathcal{D}' = \mathcal{D}_1 \circ \mathcal{D}_2$.

Lastly, it remains to verify the following:

1. Let $B_1, \ldots, B_k$ be the list of admissible backgrounds for the encoder $\mathcal{E}_1$. Then, $\mathcal{E}_2(B_1), \ldots, \mathcal{E}_2(B_k)$ is also a list of admissible backgrounds; this follows from the fact that $\mathcal{E}_2$ is a cellular transformaton.

2. $\mathcal{E}'$ is a mapping implementable by a Turing machine in the sense of Definition D.5 with admissible backgrounds $\mathcal{E}_2(B_1), \dots, \mathcal{E}_2(B_k)$. This follows from the fact that $\mathcal{E}_2$, as a cellular transformaton, can be implemented by a Turing machine.

3. $\mathcal{D}'$ and τ' can be implemented by a Turing machine in the sense of Definition D.5. This again follows from the fact that $\mathcal{D}_2$ as a cellular transformaton, can be implemented by a Turing machine.

□

An important corollary of the previous lemma is the following relationship between local and global universality.

Theorem D.29. *Each globally universal CA is also locally universal.*

Proof. It is apparent that there exists a CA $\mathcal{A}_d = (S^{\mathbb{Z}^d}, F)$ which can simulate a universal Turing machine $\mathcal{T}_U$ for every dimension $d \in \mathbb{N}$. Let $\mathcal{B} = (T^{\mathbb{Z}^d}, G)$ be a globally universal CA in dimension $d \in \mathbb{N}$. Since $\mathcal{B}$ can simulate $\mathcal{A}_d$, using Lemma D.28 we get that $\mathcal{B}$ can also simulate $\mathcal{T}_U$ and thus that $\mathcal{B}$ is also locally universal. □

In what follows, we show that global universality is a strictly stronger condition than local universality in every dimension. To be precise, the following statement holds:

Theorem D.30. *For each $d \in \mathbb{N}$ there exists a d-dimensional cellular automaton which is locally universal but is not globally universal.*

The argument is relatively simple, and uses the known fact that reversible cellular automata can be locally but not globally universal. We note that a more technical and general version of this statement was presented in [27]. We begin by introducing reversible automata.

A cellular automaton with global rule $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is reversible if F is a bijection. In this case, since F^{-1} is continuous and commutes with all shifts, the Curtis-Hedlund-Lyndon Theorem B.2 implies that it is also the global rule of a CA. While there is an algorithm that decides the reversibility of each 1D CA, this problem is proven to be undecidable for dimensions $d \geq 2$ by Kari [104]; a general overview of results regarding reversible CAs is [24]. Reversible CAs have been widely studied in the context of their computational capabilities; we list a selection of results [29, 105, 71, 89].

Theorem D.31 (Reversible CAs are locally universal [70, 28])**.** *For every $d \in \mathbb{N}$ there exists a d-dimensional reversible cellular automaton which is locally universal.*

Proof. Toffoli famously showed that for any d-dimensional CA can be simulated by a $d+1$-dimensional reversible CA [70], which proves the result for $d \geq 2$. Later, in 1995, Morita showed the result for $d = 1$, [28]. To be precise, one has to verify the constructions from the two works are compliant with our Definition D.5 of local simulation, which is a straightforward yet technical exercise. □

We proceed by showing that any reversible CA is limited in terms of what it can simulate *globally*.

Theorem D.32 (Reversible CAs cannot be globally universal)**.** *Let $\mathcal{R}$ be a d-dimensional reversible CA with global rule $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ and let $\mathcal{A}$ be a CA with states $\mathbf{2} \coloneqq \{0, 1\}$ whose global rule F is a constant mapping sending everything to $0^{\mathbb{Z}^d}$. Then, $\mathcal{R}$ cannot globally simulate $\mathcal{A}$.*

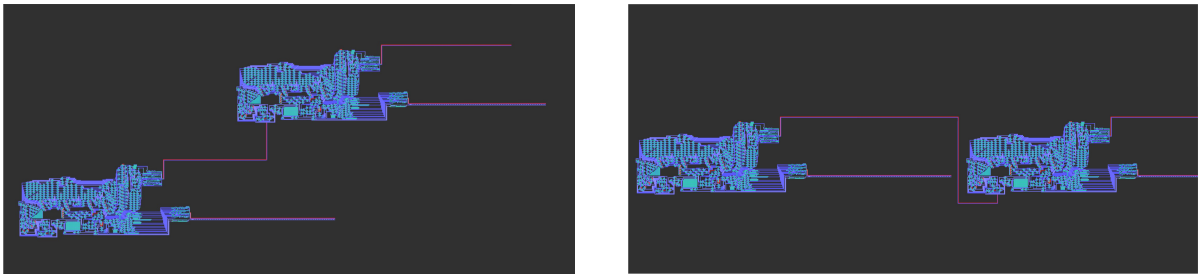
Proof. For contradiction, we will assume that there exists a shift vector $v \in \mathbb{Z}^d$, a time delay $\tau \in \mathbb{N}$, an encoder $\mathcal{E} : \mathbf{2}^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$, and a decoder $\mathcal{D} : S^{\mathbb{Z}^d} \rightharpoonup \mathbf{2}^{\mathbb{Z}^d}$ witnessing that $\mathcal{R}$ globally simulates $\mathcal{A}$. In particular, since $\mathcal{E}$ satisfies the conditions from Definition C.4, there exist sequences $V = (v_1, \ldots, v_d)$ and $W = (w_1, \ldots, w_d)$ of linearly independent vectors from $\mathbb{Z}^d$ such that $\sigma_{w_i} \circ \mathcal{E} = \mathcal{E} \circ \sigma_{v_i}$ for all $i \in \{1, \ldots, d\}$. We list a series of observations:

1. Since $\mathcal{R}$ is reversible, G is injective, and $\sigma_v \circ G^\tau$ is also injective.
2. Because of the condition $\mathcal{D} \circ \mathcal{E} = \mathrm{id}_{\mathbf{2}^{\mathbb{Z}^d}}$, $\mathcal{D}$ cannot be a constant mapping (this is important to see that $\mathcal{D}$ cannot be the one computing the constant function of $\mathcal{A}$).
3. Let $c \in \mathbf{2}^{\mathbb{Z}^d}$ be a V-periodic configuration. Then, $\mathcal{E}(c)$ is a W-periodic configuration. We denote by S_W the set of all W-periodic configurations in $S^{\mathbb{Z}^d}$. Since W spans a d-dimensional sublattice of $\mathbb{Z}^d$, the set S_W is finite.
4. Clearly, the mapping $\sigma_v \circ G^\tau$ preserves the W-periodicity of a configuration, since it is a CA.

Now, we can finish the proof. Let $c \in \mathbf{2}^{\mathbb{Z}^d}$ be an arbitrary V-periodic configuration such that $c \neq 0^{\mathbb{Z}^d}$. Since $\sigma_v \circ G^\tau\big|_{S_W} : S_W \to S_W$ is an injective mapping on a finite set, there exists some $0 < n < |S_W|$, such that $(\sigma_v \circ G^\tau)^n \circ \mathcal{E}(c) = \mathcal{E}(c)$. This gives the contradiction since on one hand $\mathcal{D} \circ \mathcal{E}(c) = c \neq 0^{\mathbb{Z}^d}$ and on the other hand $\mathcal{D} \circ \mathcal{E}(c) = \mathcal{D} \circ (\sigma_v \circ G^\tau)^n \circ \mathcal{E}(c) = F^n(c) = 0^{\mathbb{Z}^d}$. □

E One-dimensional universal self-replicator

In von Neumann's construction of a universal self-replicator [1], the new copy of the machine is built with a vertical offset. Our goal is to modify this construction so that the copies are built at the exact same height as the original machine. This will allow us to compress the dynamics into a single dimension, resulting in a 1D CA capable of universal self-replication. The general idea is illustrated in Fig. 23.



(a) (b)

Figure 23: General idea behind the self-replicator modification. (a) Scheme of the original universal self-replicator, as implemented by Buckley [106] in the software Golly [35]. (b) Illustration of the modified self-replicator we propose that has no vertical offset.

Concretely, we will modify the construction of J. W. Thatcher [31] who provides a simplification of von Neumann's universal self-replicator.

E.1 Overview of the original construction

The Von Neumann Self-Replicator (VNSR) consists of three functional blocks: the supervisory unit, the tape unit, and the constructing unit; as illustrated in Thatcher's Fig. 24. The tape unit

is responsible for reading binary data from the tape (which is indexed by positive integers, and extends to the right) and transmitting the corresponding signals to the information channel. The constructing unit operates a constructing arm which can move around the plane and, on the basis of incoming inputs, construct an arbitrary array of (inactive) cell states. The supervisory unit represents the "universal computer" that communicates via the information channel both with the tape unit – giving the reading head instructions to read a symbol or move; and with the constructing unit – sending instructions to the construction arm to move to a certain location and to create a new state there. The supervisory unit consists of components which represent internal states of the machine and also encode transitions between the states based on current state and potentially, on the tape symbol read.

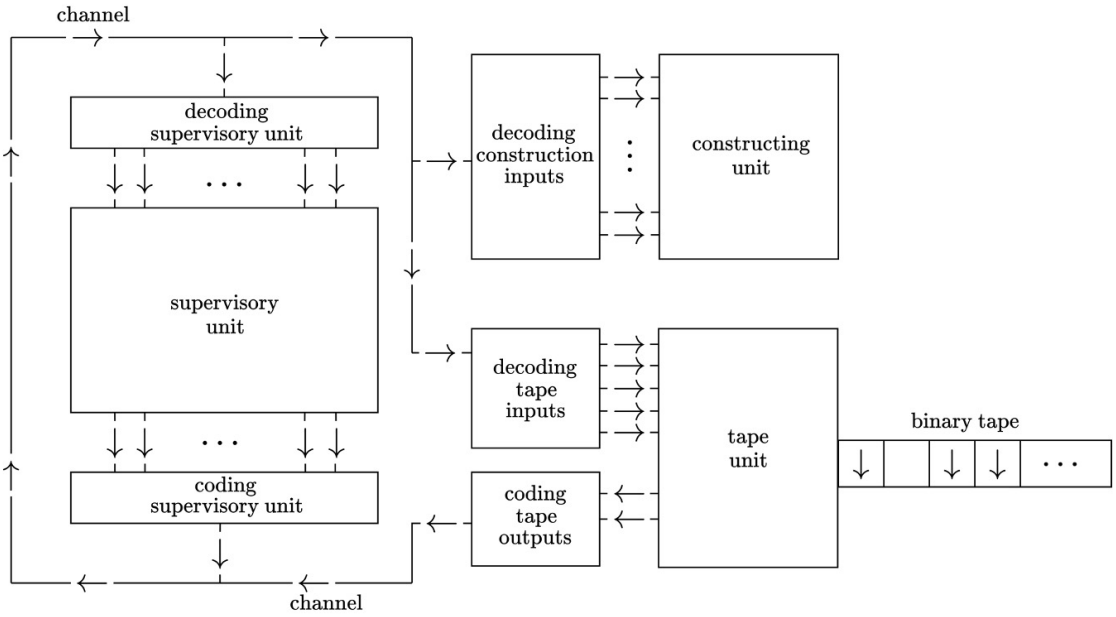


Figure 24: General organization of the VNSR adapted from [31].

The original VNSR can be made to have the following modes of operation:

1. **Computation:** VNSR performs computation based on its tape instructions. If it halts, we assume the reading head returns to the first symbol of the tape and VNSR moves to phase 2.
2. **Self-Replication:** the VNSR scans the tape; upon recognizing the self-replication instructions it starts the self-replication process.

The construction arm can move upwards and to the right from its original position, and can also retract to the left and downwards. As such, it can operate in a quadrant of the plane where the VNSR's copy is built, as illustrated in Fig. 25. This creates a vertical and horizontal offset, as illustrated in Fig. 23a. Our goal is to modify Thatcher's construction such that we get rid of the vertical offset, which will result in creating the "offsprings" at the exact same height as the original machine.

E.2 Modification for the one-dimensional setting

We modify the original design so that at the beginning of the machine's self-replication phase, the constructing arm has been prolonged so that its initial position is shifted "right" and "downward"

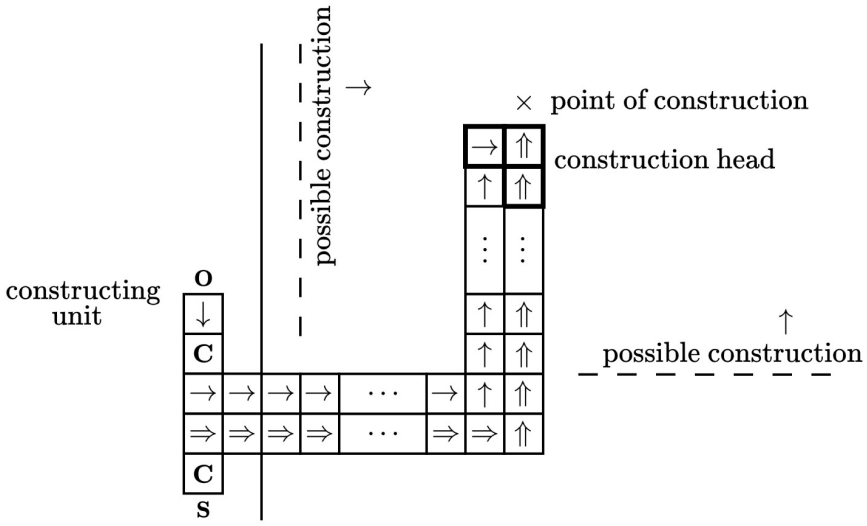


Figure 25: Spatial restriction of the construction arm's movement, adapted from [31].

to allow the newly constructed machine to be at the exact same height as its "parent machine". Our design of the universal self-replicator will have the following modes of operation:

1. **Computation:** Analogous to the original.
2. **Arm extension:** The tape reader reads each tape symbol one by one, progressively moving to the right, upon reaching and recognizing a stop sequence ω marking the end of the non-empty tape. For each symbol read, the modified supervisory unit instructs the constructing arm to be prolonged by one cell to the right. Once the process is over, we have a guarantee that the constructing arm's head is far enough to the right not to interfere with the tape's content. Afterwards, a special sequence is emitted to the construction arm that prolongs the arm downwards by a constant number of cells, ensuring the constructing arm's new initial position is at the exact height needed.
3. **Self-replication:** Analogous to the original.

Concretely, the modification we suggest is quite straightforward and follows the strategy below:

- Lemma E.1: The supervisory unit can be modified to accommodate for the extension of the construction arm in Phase 2.
- Lemma E.2: The construction arm can be modified so that at Phase 2., it is possible for it to move to the right and downwards.
- Lemma E.4 The modifications summarized by the lemmas above result in a modified VNSR whose height is bounded by 1000.

Below, we present the strategy in more detail.

Lemma E.1. *The supervisory unit of the VNSR can be modified in such a way that after the computation phase, the VNSR enters a new phase called the extension phase. During this phase, the supervisory unit uses the tape reader to scan the whole tape, and for each symbol, the supervisory unit sends a specific sequence σ to the construction unit. Subsequently, the modified supervisory unit returns the reading head to the first tape symbol.*

Proof. We will assume the beginning of the tape is delimited by the sequence ω_1 and the end of the non-empty tape is delimited by ω_2; similar to the original construction by von Neumann [1] p. 115. Further, we assume the reading head is positioned on the first tape symbol.

Thatcher's architecture of the supervisory unit is shown in Fig. 26. Each component C_i contains a decoder which, upon recognizing the encoding of its corresponding index i in the channel, activates the component C_i. Upon being activated, the component C_i is able to either:

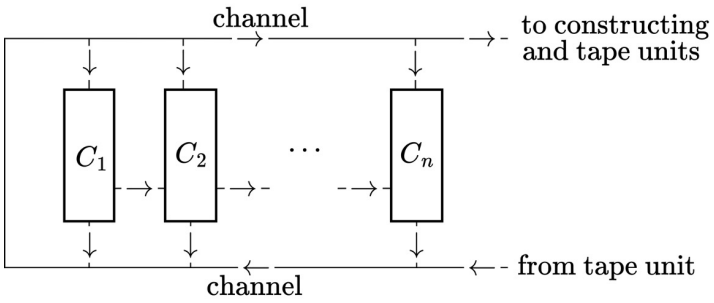


Figure 26: Scheme of the internal organization of the supervisory unit.

- Send an instruction to the tape unit to read the next symbol, and conditional on the result, broadcast the index j of the next component that should be activated.
- Send an instruction to the construction arm, and activate the consecutive unit with index $i + 1 \bmod n$.

We propose adding three more components, C_{n+1}, C_{n+2}, C_{n+3} where:

- C_{n+1} is activated after the computation phase is finished, and prompts the reading head at position i to read k consecutive symbols from the tape $a_1 \dots a_k$ where $k = |\omega_2|$, and to return to the tape index $i + 1$. C_{n+1} contains a recognizer deciding whether $a_1 \dots a_k = \omega_2$ marks the end of the tape. If not, C_{n+1} activates the component C_{n+2}. If yes, it activates C_{n+3}.
- C_{n+2} sends the sequence σ to the construction unit, which encodes the instruction to extend the construction arm one step to the right. Then, it activates C_{n+1}.
- C_{n+3} emits a signal to the construction unit with the instruction to extend the construction arm k_1 steps to the right, then k_2 steps downwards and finally, three steps to the right (where k_1 is a constant ensuring the arm will not interfere with the original machine's tape and k_2 is a constant corresponding to the height of the original machine). Afterwards, C_{n+3} activates the component C_1 which proceeds with returning the reading head to the first tape symbol and with the self-replication phase.

Each component C_i consists of a limited number of the following units:

- the recognizer $R(\underline{u})$ recognizing the encoding of sequence u form the information channel: either to recognize the sequence $\underline{i}$ marking the activation of the component C_i or to recognize the tape symbols currently read) (each of height 7; [31] p. 33)
- the pulser $P(\underline{u})$ which upon activation outputs a series of instructions u into the channel: either for the construction arm or for the reading head (each of height 3; [31], p. 28)

- A periodic pulser $PP(1)$ which ensures that an activated component C_i stays activated while waiting for the tape symbols being read (height 6, [31], p. 35)
- A delay that ensures proper operation of the tape and constructing units (height at most 300; [31], p. 95)

This allows us to bound the height of the supervisory unit by 500. □

Lemma E.2. *At the beginning of the extension phase, the construction arm's initial position can be modified so that the arm can grow down and to the right and afterwards, can create the arm's head to the right of the original VNSR's end of the tape.*

Proof. We modify the initial configuration of the constructing arm from the original Fig. 27 a) to the new Fig. 27 b).

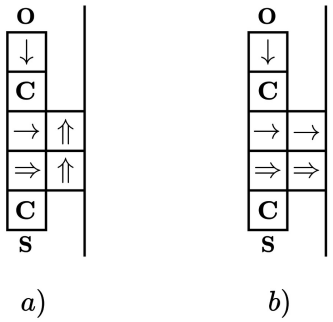


Figure 27: a) Original initial configuration of the constructing arm. b) Modified initial configuration that will result in a shifted constructing arm.

Once the extension process starts, we assume that each non-blank symbol on the tape being read gets converted by the control unit to a signal sequence σ arriving at the constructing unit into a pulser generating the sequence 10000 for the ordinary transmission channel input **O** and the sequence 1011 for the special transmission channel input **S**, resulting in the arm being prolonged by one cell to the right. After all the non-blank tape symbols are read, the state of the constructing arm is shown in Fig. 28 a). Once the whole tape is read, a special signal is transmitted to the construction unit, encoding the final horizontal shift of the constructing arm by a constant number of cells, and subsequently prolonging it by a constant number of arrows downwards and to the right, as depicted in Fig. 28 b). This ensures that the new machine will be built at the exact same height as its parent.

One important aspect of the original construction is the following: if the ordinary sequence of instructions is $i_1 i_2 \dots i_m$ and the special sequence is $j_1 j_2 \dots j_m$, then i_1 and j_1 will arrive at **O** and **S** at the same time. In addition, i_1 and j_1 will also arrive at the corresponding inputs to the constructing arm's head at the same time. We note that this confluence is preserved for the shifted constructing arm, as can be easily verified from Fig. 28 b). This allows for the rest of the self-replication process to be logically organized in exactly the same way as in the original. □

Remark E.3. *In most constructions, the 'coding' part of the tape is read-only or otherwise not extended during self-replication. In this case, the length that we need to extend the arm is the length of the coding part of the tape plus a fixed constant. Otherwise in constructions for which the coding part of the tape is manipulated during self-replications and is extended to a constant multiple of*

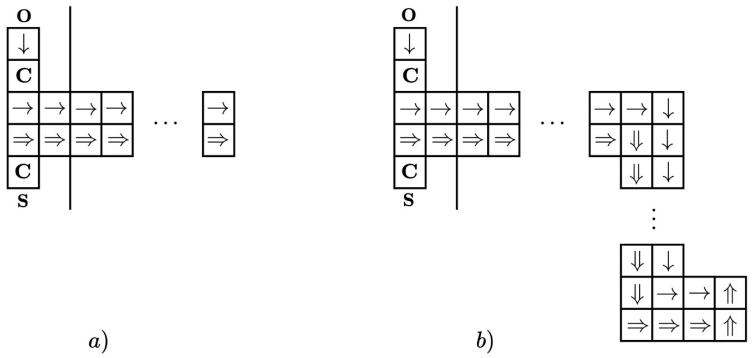


Figure 28: a) State of the constructing arm after all non-blank symbols are read; the length of the arm is proportional to the length of the tape. b) Final state of the shifted constructing arm.

its original length, this is readily accommodated by extending the arm to a constant multiple of the coding tape length plus a universal constant. (If the coding region is extended in a non-linear fashion this can also be readily accommodated in the extension of the arm.)

Lemma E.4. *The VNSR resulting from the modification of previous lemmas stays within a strip of height at most 2000.*

Proof. First, we discuss an upper bound for the original VNSR's height. In [31], Thatcher specifies the following sizes:

- tape control unit: height of 83 units, length of 107 units
- constructing unit: height of 33 units, length of 139 units

Though the supervisory unit's size is not specified, we discussed its upper bound to be 500 in Lemma E.1. Thus, the original VNSR's height can be upper bounded by 1000.

Lemma E.1 requires the modification of the supervisory unit by adding extra components. As those are chained horizontally, this does not change the supervisory unit's height. Since the newly added components contain a few extra pulsers and recognizers to accommodate for their extended functionality, we propose an upper bound for the modified supervisory unit's height to be 1000. Lemma E.2 does not require any change of the internal organization of the constructing unit, apart from the modification of the initial constructing arm state, which does not alter its height.

To summarize, a generous upper bound for the height of the modified VNSR that we propose is 2000. □

Theorem E.5. *There exists a 1D universal self-replicator.*

Proof. From Lemma E.4 we have the existence of a 2D CA with 29-states (the von Neumann CA) and an initial pattern of height 2000 which represents a universal self-replicator. From the construction, it is apparent that during the whole process of self-replication, any cell outside of the strip of height 2000 stays within the quiescent state. Thus, we can define a 1D CA with 29^{2000} states and nearest neighbors where:

- each state represent a vertical column of height 2000 in the von Neumann CA;
- the local rule $f(a, b, c)$ mimics one step of the von Neumann CA applied to a 2000×3 rectangle with columns represented by a, b, and c surrounded by the quiescent state, and outputs the outcome in the state corresponding to the middle column of the updated rectangle.

□

F Local self-replicating condition and its generalizations

In this section, our goal is to formalize necessary conditions for universal self-replication. The conditions we provide are by no means sufficient; indeed, giving a formal characterization of non-trivial self-replicating CAs is a challenging open problem. However, these conditions will be very useful for the next section where we show the existence of a locally universal CA that cannot be universally self-replicating.

Informally, we first identify which CA patterns could be perceived as "organisms": very loosely, an organism is any sequence of finite patterns that define the organism's canonical activity. For instance, the most classic glider in Game of Life periodically traverses four different shapes as it moves through the space; see Fig. 29.

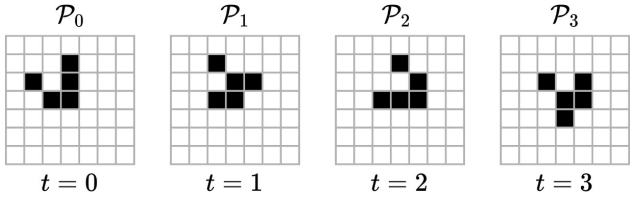


Figure 29: A classical glider in Game of Life traverses four basic shapes $(\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3)$ as it moves.

We say that such an "organism" is self-replicating, if throughout the CA's iterations, it generates arbitrarily many copies of itself. To exclude cases of trivial patterns, such as a single black cell on a white background growing progressively in all directions, we further require the "organism" to contain a component that has a temporal period of length at least two. To further build the intuition behind non-trivial self-replication, we introduce two famous examples from the literature.

F.1 Examples of non-trivial self-replication

Example F.1 (Langton's loop)**.** *Langton's loop [4] is a seminal construction of a non-universal self-replicating CA structure which is nevertheless non-trivial. The loop consists of an external sheath colored red in Fig. 30. Inside the sheath there is a circulating sequence of states encoding the loop itself. Whenever there is enough space next to the loop, the whole pattern self-replicates. Once the loop runs out of space, it "dies out" and becomes a static pattern.*

Example F.2 (Byl's loop)**.** *Byl's loop [5] is a further simplification of Langton's construction. By removing the inner sheath, Byl reduced the loop's size from 86 to 12, and the replication period from 151 to 25. Once the loop runs out of surrounding space, it enters a temporal period of size 4.*

We also comment on:

Example F.3 (von Neumann's universal self-replicator)**.** *As previously discussed, von Neumann's universal self-replicator [1] can be programmed to replicate itself (and then subsequently perform additional computation, if these instructions are appended to the self-replication instructions). Then the subsequent copies of the machine will likewise self-replicate in an identical fashion.*

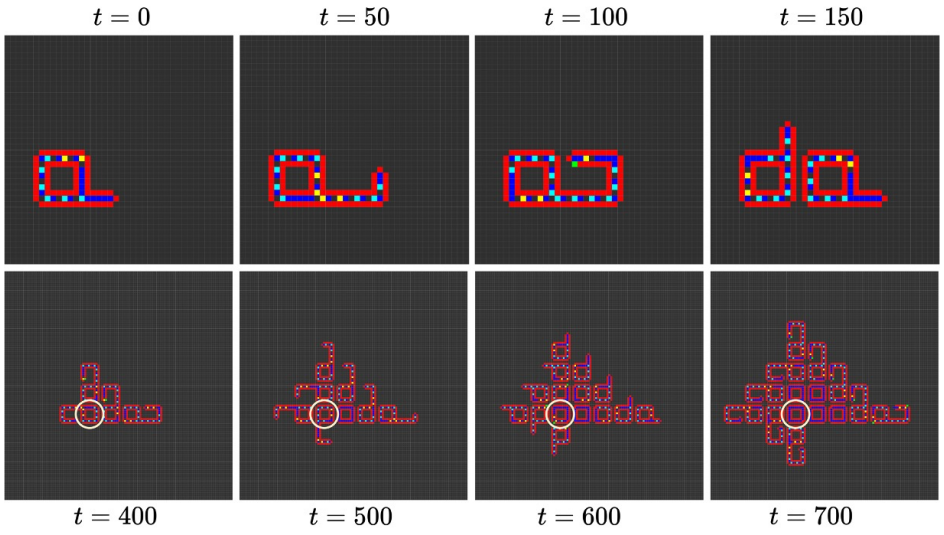


Figure 30: Langton's loops [4]. (First row) Initial process of self-replication. (Second row) Original loop is highlighted by a circle, and eventually becomes static.

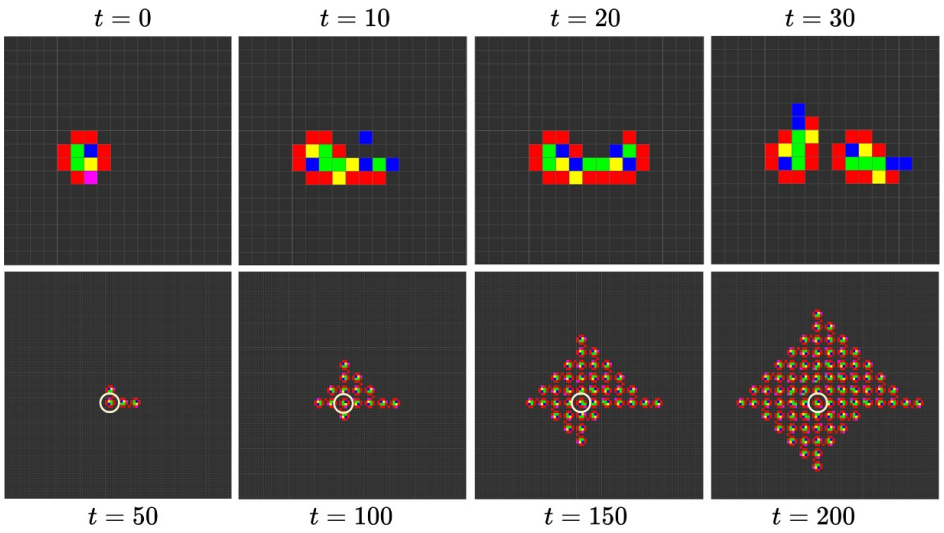


Figure 31: Byl's loops [5]. (First row) Initial process of self-replication. (Second row) Original pattern is highlighted by a circle, it eventually reaches a temporal cycle of size 4.

F.2 Formulating necessary conditions for self-replication

First, we say that each "CA organism" is in part characterized by a sequence of patterns $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{T-1})$ which describe its "canonical form of activity". In order for the organism to self-replicate, this activity (e.g. the regular motion of a subsystem of the organism, or even the dynamics of self-replication itself) has to be present in a growing number of disjoint CA regions throughout the CA's iterations.

For this, we have to address the issue of how to count the number of an organism's copies in a CA configuration. To account for various examples of non-trivial self-replication, it becomes clear that one should allow basic geometrical transformations, such as rotations, when identifying two organisms as identical. This becomes apparent, for instance, from the example of Langton's loops. We formally define this below.

Definition F.4 (Equivalence of local patterns)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a cellular automaton, $R \subset \mathbb{Z}^d$ a finite region, and $T \in \mathbb{N}$. Consider $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{T-1})$ a sequence of pairwise distinct R-patterns; i.e. for each $0 \leq i < T$ we have $\mathcal{P}_i : R \to S$. We will refer to $\boldsymbol{\mathcal{P}}$ as a local pattern sequence with temporal length T, or just a local pattern.*

Let $R' \subset \mathbb{Z}^d$ and let $\boldsymbol{\mathcal{P}}' = \left(\mathcal{P}'_0, \mathcal{P}'_1, \ldots, \mathcal{P}'_{T-1}\right)$ be another sequence of disjoint R'-patterns with the same temporal length T. We say that $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{P}}'$ are equivalent, if there exists an isometry $\varphi : \mathbb{Z}^d \to \mathbb{Z}^d$ and an integer $0 \leq \sigma < T$ such that:

$$\mathcal{P}_i = \mathcal{P}'_{\sigma + i \bmod T} \circ \varphi \qquad \text{for all } 0 \leq i < T.$$

Let $c \in S^{\mathbb{Z}^d}$ be a configuration of $\mathcal{A}$ and let $k \geq 1$ be an integer. We say that c contains k copies of $\boldsymbol{\mathcal{P}}$ if there exist k disjoint regions $R_1, \ldots, R_k \subset \mathbb{Z}^d$ such that $(c\big|_{R_i}, F(c)\big|_{R_i}, \ldots, F^{T-1}(c)\big|_{R_i})$ defines a local pattern equivalent to $\boldsymbol{\mathcal{P}}$ for all $1 \leq i \leq k$. We say that c contains exactly k copies of $\boldsymbol{\mathcal{P}}$, if it contains k copies of $\boldsymbol{\mathcal{P}}$ and it does not contain $k+1$ copies of $\boldsymbol{\mathcal{P}}$.

Now we are ready to present the definition of a self-replicating local pattern.

Definition F.5 (Self-replicating local pattern)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a cellular automaton, $R \subset \mathbb{Z}^d$ a finite region, and $T \in \mathbb{N}$. Consider $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{T-1})$ a local pattern. Let $c \in S^{\mathbb{Z}^d}$ be a configuration of $\mathcal{A}$. We say that $\boldsymbol{\mathcal{P}}$ self-replicates with configuration c if there exist two sequences of positive integers:*

$$n_1 < n_2 < n_3 < \cdots$$
$$k_1 < k_2 < k_3 < \cdots$$

such that $F^{n_i}(c)$ contains precisely k_i copies of $\boldsymbol{\mathcal{P}}$. Moreover, we require c to be a "computationally feasible" configuration; more precisely, we require that c agrees with an admissible background B (according to Definition D.4) outside of a finite region.

We note that infinite regions of an admissible background may evolve into spatially periodic, time-periodic regimes, in which a given periodic local pattern appears infinitely many times in parallel. Such patterns should not be regarded as genuine self-replicators, since they violate the requirement that the number of copies increase in a controlled manner. The following lemmas makes this precise.

Lemma F.6 (Eventual spatiotemporal periodicity of periodic configurations)**.** *Let $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ be a d-dimensional cellular automaton. Suppose the initial configuration $c \in S^{\mathbb{Z}^d}$ is totally spatially periodic, i.e. there exist linearly independent vectors $v_1, \ldots, v_d \in \mathbb{Z}^d$ such that $\sigma_{v_i}(c) = c$ for all i.*

Then the trajectory of c under F eventually becomes periodic in both space and time. In particular, there exist integers $T \geq 1$ and $\widetilde{T} \geq 0$ such that

$$F^{t+T}(c) = F^t(c) \quad \text{for all } t \geq \widetilde{T},$$

and so every $F^t(c)$ remains totally spatially periodic with respect to $v_1, \ldots, v_d$ and is eventually periodic in time.

Proof. Since F commutes with translations, total spatial periodicity is preserved along the orbit: if $\sigma_{v_i}(c) = c$ for all i, then $\sigma_{v_i}(F(c)) = F(c)$ for all i. Hence the orbit of c lies inside the set K of configurations with the fixed spatial periods determined by the v_i's. Choosing a fundamental domain for this periodic lattice, each configuration in K is determined by finitely many cells, so K is finite. The restriction of F to K is therefore a self-map of a finite set, and the orbit of c in K is eventually periodic. That is, there exist integers $\widetilde{T} \geq 0$ and $T \geq 1$ with $F^{t+T}(c) = F^t(c)$ for all $t \geq \widetilde{T}$. This proves the claim. □

Lemma F.7. *Let B be an admissible background in the sense of Definition D.4, realized by a finite partition $\mathbb{Z}^d = \bigsqcup_{i=1}^k R_i$ and basic backgrounds $B_1, \ldots, B_k$ with $B\big|_{R_i} = B_i\big|_{R_i}$. Assume some R_i is unbounded and the CA dynamics maps B_i after a transient time $\widetilde{T}$ into a configuration that is totally spatially periodic and time-periodic with finite period T (as guaranteed by Lemma F.6). Let c be a configuration that agrees with B outside a finite region $Q \subset \mathbb{Z}^d$. Let $\boldsymbol{\mathcal{P}}$ be any local pattern that occurs within this time-periodic regime and reappears every T steps. Then $\boldsymbol{\mathcal{P}}$ cannot be a self-replicating local pattern for c.*

Proof. Let r be the interaction radius of the CA. Since R_i is convex and unbounded, for any $m \in \mathbb{N}$ there exists an unbounded convex subset $R_i^{(m)} \subset R_i$ which is at ℓ_∞-distance at least rm from both ∂R_i and from the finite region Q. Take $m = \widetilde{T} + \ell$ for $\ell \in \{0, 1, \ldots, T-1\}$. Then for every $t \geq \widetilde{T} + \ell$ with $t \equiv \widetilde{T} + \ell \pmod{T}$, the dynamics inside $R_i^{(m)}$ coincides with the evolution one would see if R_i were the whole lattice and there were no perturbation in Q, since no influence can reach from ∂R_i or from Q within t steps.

By hypothesis, after time $\widetilde{T}$ the restriction of B_i is spatially periodic and time-periodic with period T, so at each such time t the unbounded region $R_i^{(m)}$ contains infinitely many disjoint occurrences of $\boldsymbol{\mathcal{P}}$. Therefore, for each residue class modulo T and all sufficiently large times, the configuration contains infinitely many copies of $\boldsymbol{\mathcal{P}}$.

However, Definition F.5 requires an infinite increasing sequence $n_1 < n_2 < n_3 < \cdots$ with the configuration containing exactly $k_j < \infty$ copies of $\boldsymbol{\mathcal{P}}$ at time n_j. Since after time $\widetilde{T}$ every (large) time in each residue class contains infinitely many copies, only finitely many times remain with finitely many copies. Thus no such infinite sequence $n_1 < n_2 < n_3 < \cdots$ can exist, and $\boldsymbol{\mathcal{P}}$ is not self-replicating. □

We highlight several crucial details of Definition F.5. Our definition aims to partially capture a certain causality: the patterns present at time n_i should be the ones that give rise to the patterns at time n_{i+1}, or at least have a common progenitor or constructor. This is in contrast with the undesirable situation where infinitely many organisms are already encoded in the background c and appear at predetermined times. Our definition addresses this concern in two ways: by imposing that at time n_i there are precisely k_i organisms present (and no more than that), and by requiring the background c to be an admissible configuration. The latter requirement significantly mitigates the possibility of pre-encoded organisms, as c is simple enough not to encode a complicated infinite set of instructions to progressively generate more and more patterns throughout the CA's iterations.

In particular, Lemma F.7 shows that periodic motifs which develop in certain infinite regions of the admissible background cannot qualify as self-replicating local patterns. At last, we present the necessary condition of self-replication below. Therein, we further exclude the case of static organisms with temporal length 1.

Definition F.8 (Local self-replicating condition)**.** *We say that a cellular automaton $\mathcal{A} = (S^{\mathbb{Z}^d}, F)$ satisfies the locally self-replicating condition if there exists a local pattern $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{T-1})$ with $T \geq 2$ which self-replicates.*

Remark F.9. *It is now easy to check that both Langton's and Byl's loops satisfy the local self-replicating condition, as does von Neumann's universal self-replicator (e.g. when it is programmed to replicate).*

We conclude by providing additional justification for our definitions and highlighting their key features. The local self-replicating condition represents a biologically plausible model for non-trivial asexual reproduction, as identical organisms will have dynamical motifs in common. A crucial feature of our definition is that it does not track individual organisms or require their perpetual existence; organisms may die, and we only require that the population proliferates across arbitrary time horizons under suitable conditions in infinite volume. This infinite-volume assumption is a mathematical convenience that allows for unbounded organism proliferation, reflecting the principle that any organism should theoretically produce arbitrarily many offspring given sufficient resources. Finally, we emphasize again that our local self-replicating condition is necessarily satisfied by universal self-replicators; the reason is that their offspring will themselves replicate in an identical fashion, and so the process of replication itself forms a local pattern sequence.

These considerations suggest that more stringent necessary conditions for non-trivial self-replication could be formulated. While such additional conditions are not required for our construction of a universal CA that cannot sustain universal self-replication, we briefly discuss several possibilities below.

F.3 Comments on more general local self-replicating conditions

Our self-replicating condition imposes only mild constraints on the dynamical behavior of a would-be organism, requiring merely that it exhibit a dynamical motif comprising a local pattern of length at least two. Inspired by Langton's loops and their progressively complex generalizations, culminating in the Turing-universal Perrier-Sipper-Zahnd loops, we can envision strengthened conditions that require the local pattern to exhibit more sophisticated forms of computation. In other words, while our existing definition merely requires that an organism 'do something,' we could further demand that it 'do something interesting,' such as implementing specific forms of computation. Following our discussions of local universality in CAs, one natural extension would be to require that organisms implement computation calibrated by an encoder-decoder pair, analogous to computational dynamical systems [30]. While we do not pursue these more stringent versions of the local self-replicating condition here, we view this as a promising direction for future work, which we discuss further in Appendix H.

F.4 Universal self-replicators

As we discussed at the beginning of this section, giving characterizing conditions for non-trivial self-replication remains an open problem. Thus, we can only informally describe UniversalSelfReplicating

as the set of all cellular automata (in any dimension) that are locally universal and capable of non-trivial self-replication. We assert that any future feasible definition of non-trivial self-replication should satisfy the following properties.

Claim F.10. *Any cellular automaton belonging to the* UniversalSelfReplicating *class has to satisfy the local self-replicating condition in Definition F.8. Moreover, von Neumann's universal self-replicator has to belong to* UniversalSelfReplicating.

It is further natural to require that any CA that can globally simulate an automaton capable of non-trivial self-replication should itself be capable of non-trivial self-replication. Indeed, such CAs can simulate a universal self-replicating CA via a local encoding, as ensured by the existence of the CA specified by our Theorem E.5.

Claim F.11. GloballyUniversal $\subseteq$ UniversalSelfReplicating.

Lastly, we argue that the above inclusion is strict. Since no reversible cellular automaton can be globally universal, it suffices to prove the following.

Theorem F.12. *There exists a reversible cellular automaton belonging to* UniversalSelfReplicating*, and thus*

$$\mathsf{GloballyUniversal} \subsetneq \mathsf{UniversalSelfReplicating}\,.$$

Proof. We simply use Toffoli's result from [70], showing that for every d-dimensional CA, there exists a $d+1$-dimensional reversible CA that can globally simulate it. Thus, there exists a 3-dimensional reversible CA globally simulating von Neumann's universally self-replicating automaton. Combining previous claims, we get that this CA belongs to UniversalSelfReplicating. Furthermore, in the previous section, we proved the existence of a 1D CA belonging to UniversalSelfReplicating. This immediately implies the existence of a 2D reversible CA belonging to UniversalSelfReplicating. □

G Non-talking heads cellular automata

In this Section, we construct an example of a 1D CA that is locally Turing-universal and yet cannot sustain universal self-replication. This counters the intuition that local Turing universality of a CA is sufficient to imply the existence of universal self-replicating configurations. Our strategy is to first construct a nice family of locally Turing-universal CAs, and then to modify the construction so we can prove that no configuration can satisfy the periodic self-replicating condition, thereby ruling out universal self-replicating configurations. Before proceeding, we draw a conceptual connection to biological systems to clarify the rationale for our construction.

G.1 A biological analogy

In life on Earth, DNA participates in at least two distinct processes. First, DNA comprises a description of an organism and participates in replication, where that description is copied. Second, DNA enables protein synthesis via the central dogma of molecular biology: DNA makes RNA (via transcription), and RNA makes protein (via translation). These proteins, combined with ambient chemical processes, build complex biomolecular structures that serve as machines and can be viewed as facilitating or instantiating computation and information processing. The key point is that replication and protein synthesis are two distinct processes; we can imagine one without the other.

Specifically, we can envision a system capable of building protein structures but with instabilities that preclude self-replication. Indeed, many organisms, whether by mutation or other causes, lack the ability to reproduce. It is therefore not difficult to imagine a system where a judicious choice of physical laws makes self-replication not merely impractical, but impossible altogether. We pursue this line of reasoning below in a concrete example.

G.2 Absence of self-replication in non-talking heads CAs

It is first useful to recall some of our notation for Turing machines, as discussed above in Definition D.1. Let Q be a finite set of states and Σ be a finite set of symbols. Then we define the transition function by

$$\delta : Q \times \Sigma \to Q \times \Sigma \times \{\mathrm{L},\, \mathrm{R},\, \mathrm{S}\}$$

where L stands for 'left', R stands for 'right', and S stands for 'stay'. We can write out the function as

$$\delta(q, s) = (\delta_Q(q, s),\, \delta_\Sigma(q, s),\, \delta_{\text{move}}(q, s))\,.$$

Suppose that δ corresponds to a universal Turing machine. Now let us define a 1D $4(|Q|+1)|\Sigma|$-state CA as follows.

We label each state on each site of the CA by a state (h, s, a) where $h \in Q \cup \{\sqcup\}$, $s \in \Sigma$, and $a \in \{\mathrm{L}, \mathrm{R}, \mathrm{S}, \sqcup'\}$. (Here h stands for 'head', a stands for 'arrow', and $\sqcup$ and $\sqcup'$ are blank or empty.) We take one of the elements of Q to be the halting state, denoted by halt. The transition rule of the CA only depends on nearest-neighbors, and is defined as follows:

$$f((h_1, s_1, a_1),\, (h_2, s_2, a_2),\, (h_3, s_3, a_3))$$

$$= \begin{cases} (\delta_Q(h_1, s_1), s_2, a_2) & \text{if } h_1 \neq \sqcup,\ h_2 = h_3 = \sqcup, \text{ and } \delta_{\text{move}}(h_1, s_1) = \mathrm{R} \\ (\delta_Q(h_3, s_3), s_2, a_2) & \text{if } h_3 \neq \sqcup,\ h_1 = h_2 = \sqcup, \text{ and } \delta_{\text{move}}(h_3, s_3) = \mathrm{L} \\ (\delta_Q(h_2, s_2), \delta_\Sigma(h_2, s_2), \delta_{\text{move}}(h_2, s_2)) & \text{if } h_2 \neq \sqcup,\ h_1 = h_3 = \sqcup, \text{ and } \delta_{\text{move}}(h_2, s_2) = \mathrm{S} \\ (\sqcup, \delta_\Sigma(h_2, s_2), \delta_{\text{move}}(h_2, s_2)) & \text{if } h_2 \neq \sqcup,\ h_1 = h_3 = \sqcup,\ \delta_{\text{move}}(h_2, s_2) = \mathrm{L}, \text{ and } a_1 = \mathrm{R} \text{ or } \sqcup' \\ (\sqcup, \delta_\Sigma(h_2, s_2), \delta_{\text{move}}(h_2, s_2)) & \text{if } h_2 \neq \sqcup,\ h_1 = h_3 = \sqcup,\ \delta_{\text{move}}(h_2, s_2) = \mathrm{R}, \text{ and } a_3 = \mathrm{L} \text{ or } \sqcup' \\ (h_2, s_2, a_2) & \text{if } h_1 = h_2 = h_3 = \sqcup \\ (\mathsf{halt}, s_2, a_2) & \text{otherwise} \end{cases}$$

The above describes a stationary tape with moving heads, where each head records on each tape cell the direction that head is moving next. There are two additional key features: (i) if there are two or more adjacent heads, then those heads will enter the halting state; (ii) if a head is slated to move and the cell to which it is moving has an a which has an inconsistent 'parity', then that head will enter the halting state. This mechanism (ii) prevents any two heads from 'communicating' with one another. As such, we will refer to the CA as a "non-talking heads" CA. We depict the basic mechanism described here in Fig. 32.

The non-talking heads CA defined above is locally Turing-universal by construction, as recorded in the following lemma.

Lemma G.1 (Local universality of the non-talking heads CA)**.** *The 1D non-talking heads CA defined above is locally universal.*

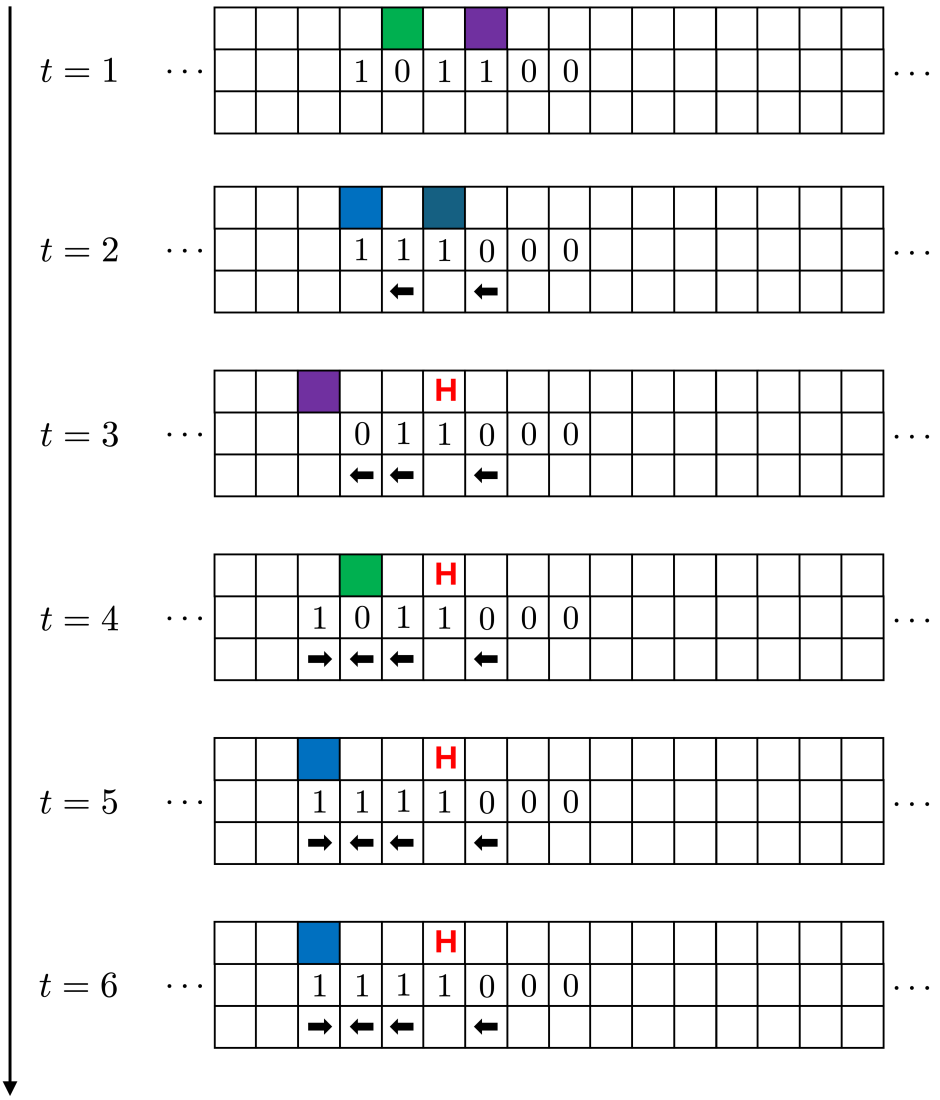


Figure 32: A depiction of the dynamics of the non-talking heads CA, with time running from top to bottom. At each moment in time, we 'unfold' a single cell into three vertical cells, corresponding to h, s, a from top to bottom. Blank cells correspond to the empty state or symbol; colored cells in the top rows correspond to heads in some particular state (with **H** denoting the halting state), 0's and 1's in the middle rows correspond to symbols on the tape, and arrows $\leftarrow$ and $\rightarrow$ in the bottom rows corresponding to $a = \mathrm{L}$ and $a = \mathrm{R}$, respectively. At time $t = 1$, we see two (activated) heads and six non-empty symbols. At time $t = 2$, the heads have moved to the left, affecting symbols and leaving arrow markers to reflect their direction of movement. At time $t = 3$ the right-most head has halted since it ran into an arrow to its left with a parity indicating that the arrow was laid down by another head. The left-most head continues to transition, affect symbols, and lay down arrows, as we see at times $t = 4$, $t = 5$, and $t = 6$.

Proof. The CA directly simulates a universal Turing machine, for example if there is only one head (i.e. only one cell is not in a state $(\sqcup, s, a)$) and the rest of the tape elements are initialized with $a = \sqcup'$. □

Now we will demonstrate that the non-talking heads CA does not have any configurations that satisfy the local self-replicating condition. To do so, we will utilize the following lemma.

Lemma G.2 (Trapping lemma for non-talking heads)**.** *Label the sites of the CA by $\mathbb{Z}$. Consider a configuration c of the CA and list the heads present at time $t = 0$ in strictly increasing order of position*

$$\cdots < h_{-2}(0) < h_{-1}(0) < h_0(0) < h_1(0) < h_2(0) < \cdots ,$$

where $h_0(0)$ labels the first head at position greater than or equal to zero in $\mathbb{Z}$. Let $h_i(t)$ denote the position at time t of the (unique) head with index i. Then for all $t \geq 0$,

$$h_{i-1}(0) < h_i(t) < h_{i+1}(0) .$$

Proof. Consider the trajectory of a particular head h_i, which we take to be the half-infinite sequence $(h_i(0), h_i(1), h_i(2), ...)$ where $h_i : \mathbb{Z}_{\geq 0} \to \mathbb{Z}$ is a map from times $t \geq 0$ to locations of cells of the CA. If $h_i(t)$ encounters either (i) another head, or (ii) an a symbol with a parity in the same direction as the transition of the head, then h_i will halt and remain at its position for all futures times t. When a head halts on account of an a which it encounters, we will say that this is an a with the 'wrong' parity relative to that head.

There are several cases to consider. The first case is that $h_i(t)$ encounters an a with the 'wrong' parity and halts before it reaches either sites $h_{i-1}(0)$ or $h_{i+1}(0)$. This can happen in two ways; either because such an a with the 'wrong' parity was encoded in the initial conditions (and not laid down by a head), or because it was laid down by a head h_{i-1} or h_{i+1}. Either situation is clearly fine, since h_i will also before reaching $h_{i-1}(0)$ or $h_{i+1}(0)$.

The second case is that $h_i(t)$ encounters either head h_{i-1} or h_{i+1} and thus halts before it reaches either sites $h_{i-1}(0)$ or $h_{i+1}(0)$. This is fine as well.

The third and final case is that $h_i(t)$ eventually encounters either site $h_{i-1}(0)$ or site $h_{i+1}(0)$. Let us consider the former, and note that the argument for the latter goes through mutatis mutandis. In the former setting, h_i must encounter $h_{i-1}(0)$ from the left. If head h_{i-1} is still present at site $h_{i-1}(0)$ then head h_i will halt at site $h_{i-1}(0) + 1$ and thus $h_{i-1}(0) < h_i(t)$ for all t. If head h_{i-1} is not present at site $h_{i-1}(0)$ then it must be to the left of that site, for if it were to the right then it would have already encountered head h_i causing both of them to halt. But then h_{i-1} would have moved left from site $h_{i-1}(0)$ and thus recorded an a there which has the wrong parity relative to h_i; as such, if h_i attempts to transition leftward into site $h_{i-1}(0)$, then h_i will halt at site $h_{i-1}(0) + 1$, again giving $h_{i-1}(0) < h_i(t)$ for all t. Nearly identical arguments likewise give us $h_i(t) < h_{i+1}(0)$ for all t, leading to the desired claim. □

With the above lemma at hand, we can now prove the main theorem of this section.

Theorem G.3 (Non-talking heads CA does not have self-replication)**.** *The 1D non-talking heads CA does not have any configurations satisfying the local self-replicating condition. Thus the CA is in* LocallyUniversal *but not in* UniversalSelfReplicating*, implying*

$$\mathsf{UniversalSelfReplicating} \subsetneq \mathsf{LocallyUniversal} .$$

Proof. By contradiction, suppose that c is a configuration of the CA satisfying the local self-replicating condition. Since by hypothesis c is a self-replicating pattern, it agrees with an admissible background B outside of a finite region. Without loss of generality, we decompose $\mathbb{Z} = R_- \sqcup R_0 \sqcup R_+$ where $R_- = (-\infty, a-1] \cap \mathbb{Z}$, $R_0 = [a, b] \cap \mathbb{Z}$, and $R_+ = [b+1, \infty) \cap \mathbb{Z}$ for some integers $a < b$. We suppose that c equals $B_1\big|_{R_-}$ on R_-, a finite pattern Q on R_0, and $B_2\big|_{R_+}$ on R_+, where B_1, B_2 are basic backgrounds.

Let $\widetilde{T}_1$ be the time that it takes for B_1 to evolve to its steady state (which is guaranteed to exist by Lemma F.6), which is necessarily spatially periodic and time-periodic. We similarly let $\widetilde{T}_2$ be the time that it takes for B_2 to evolve to its steady state (again guaranteed to exist by Lemma F.6), and take $\widetilde{T} := \max\{\widetilde{T}_1, \widetilde{T}_2\}$. Further let $\min Q$ denote the location of the left boundary of Q, and let $\max Q$ denote the location of the right boundary of Q. We denote

$$S_t^- := (-\infty, \min Q - t - 1] \cap \mathbb{Z}$$
$$S_t^+ := [\max Q + t + 1, \infty) \cap \mathbb{Z}$$

which captures regions untouched by the forward lightcone of Q at time t. Then for all $t \geq \widetilde{T}$,

$$F^t(c)\big|_{S_t^-} = F^t(B_1)\big|_{S_t^-}, \quad F^t(c)\big|_{S_t^+} = F^t(B_2)\big|_{S_t^+},$$

reflect the steady state dynamics of B_1 and B_2, respectively. Let us consider the right-most head in the half-infinite configuration $F^{\widetilde{T}}(c)\big|_{S_t^-}$, and let $a' - 1$ be the right-most coordinate that it reaches in the steady-state dynamics of B_1. Similarly, let us consider the left-most head in the half-infinite configuration $F^{\widetilde{T}}(c)\big|_{S_t^+}$, and let $b' + 1$ be the left-most coordinate that it reaches in the steady-state dynamics of B_2. It will be useful to consider the decomposition $\mathbb{Z} = R'_- \sqcup R'_0 \sqcup R'_+$ where $R'_- = (-\infty, a' - 1] \cap \mathbb{Z}$, $R'_0 = [a', b'] \cap \mathbb{Z}$, and $R'_+ = [b' + 1, \infty) \cap \mathbb{Z}$.

Next we consider the dynamics of $F^t(B_1)\big|_{R'_+}$ for $t \geq \widetilde{T}$. Since the dynamics in the region R'_+ and at times $t \geq \widetilde{T}$ is periodic in both space and time, we can decompose $R'_+ = R'^{(1)}_+ \sqcup R'^{(2)}_+ \sqcup R'^{(3)}_+ \sqcup \cdots$ where each $R'^{(i)}_+$ has the same length and evolves identically. Moreover, we can choose the $R'^{(i)}_+$ so that each has a fixed number of heads which never leave that region. As such, the $R'^{(i)}_+$ are *non-interacting*: affecting the dynamics of any one $R'^{(i)}_+$ does not affect the dynamics of any others. We can perform a similar decomposition $R'_- = R'^{(1)}_- \sqcup R'^{(2)}_- \sqcup R'^{(3)}_- \sqcup \cdots$ by an analogous argument.

For times $t \geq \widetilde{T}$, the dynamics in the region R'_0 can at most affect $R'^{(1)}_-$ and $R'^{(1)}_+$, causing some of their heads to halt. But the regions $R'^{(i)}_-$ and $R'^{(i)}_-$ for all $i \geq 2$ will be unaffected, since they do not interact with $R'^{(1)}_-$ and $R'^{(1)}_+$, or with R'_0. Therefore

$$F^t(c)\big|_{R'_- \setminus R'^{(1)}_-} = F^t(B_1)\big|_{R'_- \setminus R'^{(1)}_-}, \quad F^t(c)\big|_{R'_+ \setminus R'^{(1)}_+} = F^t(B_2)\big|_{R'_+ \setminus R'^{(1)}_+}, \quad \text{for all } t \geq \widetilde{T}.$$

If $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_{T-1})$ is the self-replicating pattern, letting L be the length of the region on which the pattern is supported. Then either a copy of $\boldsymbol{\mathcal{P}}$ is fully within $R'_- \setminus R'^{(1)}_-$ or $R'_+ \setminus R'^{(1)}_+$, or otherwise it is fully within $[a' - L, b' + L]$. In the former setting, $\boldsymbol{\mathcal{P}}$ is background-borne and excluded by Lemma F.7, since infinitely many copies of it will arise at times $t \geq \widetilde{T}$.

Finally, we note that at any one time, there can only be finitely many disjoint copies of $\boldsymbol{\mathcal{P}}$ within the finite region $[a' - L, b' + L]$. But this means that the total number of disjoint $\boldsymbol{\mathcal{P}}$'s is bounded for all time, and therefore cannot replicate beyond a certain number. Thus we reach a contradiction. □

G.3 Self-replication hierarchy for CAs

We now have all the results in place to establish a self-replication hierarchy for CAs. By combining Theorem F.12 with Theorem G.3, we have:

Theorem G.4 (Self-replication hierarchy for CAs)**.** *We have the strict inclusions*

$$\mathsf{GloballyUniversal} \subsetneq \mathsf{UniversalSelfReplicating} \subsetneq \mathsf{LocallyUniversal}\,.$$

This is one of the main results of our paper. We emphasize that this stratification clarifies both the relationship between global and local universality and their connection to (universal) self-replication. The natural intuition from von Neumann's work on universal self-replicators [1], that any system capable of furnishing local Turing-universal computation is also capable of (perhaps universal) self-replication, is evidently false. However, in establishing our counterexample, we have been forced to develop sharper mathematical criteria for when the dynamics of physical systems constitute non-trivial self-replication, and to identify dynamical mechanisms that are essential for such self-replication to arise.

H Open problems and conjectures

We conclude by articulating a number of open problems and conjectures motivated by our work. In so doing, we sketch the contours of a research program for establishing a general theoretical foundation for self-replication beyond particular examples. We will first give some more concrete questions before delving into grander goals.

Effect of dimensionality of CAs. In our discussion of $\mathsf{UniversalSelfReplicating}$ we took the set of CAs to be a union over all dimensions. That is, letting $\mathsf{UniversalSelfReplicating}_d$ be the set of d-dimensional CAs exhibiting universal self-replication, we have $\mathsf{UniversalSelfReplicating} := \bigcup_{d=1}^{\infty} \mathsf{UniversalSelfReplicating}_d$. It is natural to ask questions about each $\mathsf{UniversalSelfReplicating}_d$ individually.

Our example of a $d = 1$ version of von Neumann's universal self-replicator demonstrates that $\mathsf{UniversalSelfReplicating}_1$ is non-empty. The CA in question is not reversible (i.e. it is not in $\mathsf{Reversible}$). However, we can establish the following result for higher dimensions:

Corollary H.1 (to Theorem 2.5)**.** *There are reversible CAs in any dimension $d \geq 2$ which are universally self-replicating, or equivalently* $\mathsf{UniversalSelfReplicating}_d \cap \mathsf{Reversible}$ *is non-empty for $d \geq 2$.*

This follows from the well-known fact that a CA in $d = 1$ can be embedded in a reversible CA in any higher dimension. Thus, we can use our $d = 1$ version of von Neumann's construction to exhibit a reversible CA in $d \geq 2$ dimensions which is universally self-replicating. For the 1D case, we conjecture:

Conjecture H.2. *There is a reversible CA in one dimension which is universally self-replicating, or equivalently* $\mathsf{UniversalSelfReplicating}_1 \cap \mathsf{Reversible}$ *is non-empty.*

The most natural way to establish this conjecture would be to build a reversible 1D universal self-replicator 'from scratch'. However, one might be tempted to adapt higher-dimensional constructions. One route would be to construct a reversible 2D universal self-replicator, and then to show that it can both perform universal computation and self-replicate in a strip of bounded height

(although this might be harder to achieve in the reversible setting than the irreversible setting). Then one could adapt the strategy of Theorem E.5 to 'compress' the reversible 2D CA into a reversible 1D CA.

Another tempting approach would be to take our construction of a 1D universal self-replicator in Theorem E.5, and try to 'simulate' it by a reversible 1D CA. However, this route appears to be difficult, or perhaps impossible, for interesting reasons. To elaborate, we note that there are results in the literature that any irreversible 1D CA can be simulated by a reversible 1D CA [28, 107]. Take for example the construction of [28], which shows how a reversible 1D CA can simulate any other irreversible 1D CA starting from a configuration in which all but finitely many sites are in the quiescent state. However, the 'simulator' reversible 1D CA takes $O(n)$ time steps to simulate any configuration in the irreversible CA with n non-quiescent sites, amounting to a worst-case quadratic computational slowdown. While such an encoding is viable for lifting a locally universal but irreversible 1D CA into a reversible one, it does not interface well with the local self-replicating condition. The problem is that when simulating self-replicating configurations, the simulation time for dynamical motifs increases with the number of replicator copies being simulated. From the perspective of the simulated system, the replicated organisms become 'slower' as their numbers grow. This means there is no truly preserved dynamical motif across time that is shared by all copies. (Moreover the particular simulation method in [28] can cause dynamical motifs to occur non-simultaneously when many replicator copies are present, providing another violation of the local self-replicating condition.) Other methods for simulating irreversible 1D CAs with reversible ones (see e.g. [107]) exhibit related computational slowdowns. In fact, no-go theorems such as [27] suggest that non-constant slowdowns may be unavoidable when simulating irreversible 1D CAs with reversible ones.

Turning to a new question, we can also ask whether specific CAs are universally self-replicating. To this end, we recapitulate our conjecture from earlier in the paper:

Conjecture H.3. *Rule 110 is in* $\mathsf{UniversalSelfReplicating}_1$.

An affirmative answer to this conjecture would be compelling since it would constitute by far the simplest set of CA rules giving rise to (non-trivial) universal self-replication, although the encoding would be necessarily complicated. It would suffice for Rule 110 to be globally universal (as this would imply universal self-replication on account of our $d = 1$ version of von Neumann's construction), but this is not known. The existing universality construction [15] is tailored to local universality as opposed to global universality, so it would seem that a rather different encoding would be required if global universality were to be possible. Instead, we suspect it may be possible to use (a modest generalization of) the existing local universality construction for Rule 110 to build a 1D universal self-replicator.

Genericity of universality and self-replication. Our work serves to crystallize the relationship between computational universality and non-trivial self-replication. Ultimately, we are interested in being able to diagnose if a particular CA is able to exhibit non-trivial self-replication, or perhaps more strongly universal self-replication. A sufficient condition is for the CA to be globally universal; a necessary condition for the CA to exhibit universal self-replication is for it to be locally universal. At the moment, we do not know of any other more discriminating necessary conditions, and we will discuss this later.

Do 'generic' CAs exhibit non-trivial self-replicators? Before examining this question, we take a detour to discuss a more well-studied variant. There is a belief within the CA community that 'generic' CAs exhibit computational universality of some kind. A clear articulation of this view

originates in the work of Wolfram, specifically in his work on 1D CAs [12, 108]. He offers a qualitative classification of CA behavior roughly as follows: in Class 1 CAs, almost any initial condition quickly converges to a uniform fixed point; in Class 2 CAs, dynamics settle into simple, localized periodic or stable patterns; in Class 3 CAs, activity remains aperiodic and chaotic with no persistent structure; and in Class 4 CAs, long-lived localized patterns move and interact, mixing order with apparent randomness. Wolfram's "Principle of Computational Equivalence" conjecture stipulates that Class 4 CAs are typically computationally universal [12, 108]. While conceptually stimulating, this conjecture is necessarily qualitative, as neither the Class 4 designation, computational universality, nor the notion of 'typically' are sharply defined. In practice, the conjecture is applied by identifying CAs that qualitatively belong to Class 4 and then conjecturing their computational universality, as was done with Rule 110.

We suggest that Wolfram's conjecture most naturally pertains to local universality rather than global universality. Indeed, Rule 110 is a Class 4 CA whose local universality is taken as evidence for the conjecture. More broadly, one might expect any locally universal CA to exhibit the dynamical features characteristic of Class 4. However, if Wolfram's conjecture is read as requiring that random initial conditions typically evolve into ordered, localized structures, then reversible CAs pose a principled obstacle. Because every reversible CA is surjective and (on full shifts) surjective CAs preserve the uniform Bernoulli product measure, an i.i.d. initial configuration remains i.i.d. at every fixed time and so no spatial correlations or entropy reduction are possible [109, 110]. A complementary result shows that in one dimension, surjectivity is equivalent to preserving Martin–Löf randomness, so algorithmically random configurations remain random under reversible evolution [111]. In this precise sense, reversible CAs do not generate order from randomness and therefore, under this reading, should not be counted as Class 4. (They can still display Class-4-like behavior from finite or low-entropy seeds, e.g. [112].) These considerations lead us to conjecture:

Conjecture H.4. *There exists a Class 4 CA which is irreversible and locally universal, but not globally universal.*

If true, this would support our suggestion that local universality is the natural interpretation of universality in Wolfram's conjecture. A difficulty in establishing the conjecture is that while demonstrating local universality requires finding a particular sensible encoding and decoding, ruling out global universality requires showing that no suitable encoding and decoding exist.

We now examine another aspect of Wolfram's conjecture: whether 'typical' Class 4 CAs are computationally universal. We reformulate this question to make it precise. Consider CAs on a square lattice in a fixed number of dimensions, say $d = 1$, 2, or 3. Consider CAs with interaction radius at most R, where each cell has $|S|$ possible states. We denote this set $\mathsf{CA}_{d,R,|S|}$, which has cardinality approximately $|S|^{\text{const.}\times R^d\times|S|}$. This leads to the following question:

Question H.5. *Consider the set $\mathsf{CA}_{d,R,|S|}$. For d fixed, R fixed, and $|S| \to \infty$, what fraction of the set is locally universal? What about for d fixed, $|S|$ fixed, and $R \to \infty$? What about for d fixed, and $|S|, R \to \infty$ with some ratio $f(R)/|S|$ fixed?*

Plausibly, only a small or vanishing fraction of CAs are locally universal in the first two cases. Since determining local universality is undecidable [94], a proof would need to show that generic CAs in these regimes possess obstructions to universality. This strategy has succeeded in the study of random groups, where all but a vanishing fraction have solvable word problems due to their hyperbolic geometric structure [113]. Semon Rezchikov has suggested generalizing this approach to semigroups or Turing machines; extending it to CAs would be natural.

If only a vanishing fraction of CAs (in the first two formulations of Question H.5) were locally universal, this would contradict the intuition behind Wolfram's conjecture if Class 4 CAs were

asymptotically generic. Regardless, resolving Question H.5 would provide valuable insight into the relationship between local universality and the space of CAs.

For non-trivial self-replication, the situation may be more favorable. Exhibiting non-trivial self-replication is likely weaker than local universality, and thus potentially more pervasive among CAs. As we discuss below, there may be a hierarchy of self-replication conditions that we do not yet understand; contingent on such conditions being well-defined, a variant of Question H.5 for non-trivial self-replication might optimistically yield that a large fraction of CAs support it. We pose this as a preliminary question:

Question H.6. *Do a large fraction of CAs (e.g. in some asymptotic sense of Question H.5) exhibit non-trivial self-replication?*

An affirmative answer would be particularly interesting, as it would suggest that non-trivial self-replication is in some sense generic.

Hierarchy of non-trivial self-replicating conditions. One of the contributions of our work is to articulate a necessary condition for a CA configuration to exhibit non-trivial self-replication. An essential intuition for our condition is that a self-replicating object and its replicas should be able to exhibit non-trivial dynamics (such as a periodic orbit) after replication has occurred. That is, the replicated objects are in some manner functional. It is an interesting problem to formally characterize a hierarchy of ever more stringent non-trivial self-replication conditions. We formulate this as a question:

Question H.7. *Can one define a nested sequence of dynamical conditions on CAs leading to a hierarchy*

$$\mathsf{LocallyPeriodicSelfReplicating} = \mathsf{C}_1 \subset \mathsf{C}_2 \subset \cdots \subset \mathsf{C}_n = \mathsf{UniversalSelfReplicating},$$

analogous to increasing levels of biological complexity (e.g. from virus-like copying to autonomous cell division)?

If such a hierarchy can be constructed, it would be fruitful to classify known CAs within the hierarchy.

While it is natural to organize the space of CAs according to whether they can furnish configurations whose evolutions exhibit certain dynamical properties, we could equally well classify the CA configurations themselves. For instance, given a CA rule G and a configuration c, consider the pair (G, c). So far we have been saying that G is e.g. in $\mathsf{LocallyPeriodicSelfReplicating}$ if there is at least one c whose dynamics under G satisfies the local self-replicating condition. (The viable initial c's may be restricted, such as by demanding that they vary in only a finite number of cells from a fixed background, etc.) Instead, we could consider $\mathsf{LocallyPeriodicSelfReplicating}_{\mathsf{config}}$ to be the set of pairs (G, c) such that the dynamics of c under G satisfies the local self-replicating condition. This formulation opens up new questions. For example:

Question H.8. *Let G_{Byl} be the CA rule giving rise to Byl's loops [5], and let b be its configuration of all quiescent states. As previously discussed, $G_{\mathrm{Byl}} \in \mathsf{LocallyPeriodicSelfReplicating}$. What is the best lower bound on*

$$N_{\mathrm{Byl}} = \min\left\{\mathsf{Hamming}(b, c) \,:\, (G_{\mathrm{Byl}}, c) \in \mathsf{LocallyPeriodicSelfReplicating}_{\mathsf{config}}\right\},$$

namely the number of non-quiescent states for the simplest initial configuration of G_{Byl} whose dynamics satisfies the local self-replicating condition? Byl's construction gives $N_{\mathrm{Byl}} \leq 12$.

This question pertains to a notion of *complexity* for self-replicating configurations, particularly the minimum complexity necessary to achieve a specific type of self-replicating behavior.

As remarked earlier, Langton's loops [4] and their successive generalizations to Byl's loops [5], etc., culminating in the Perrier-Sipper-Zahnd loops [33], qualitatively exhibit a hierarchy of non-trivial self-replicating behaviors. Specifically, Langton's loops belong to $\mathsf{LocallyPeriodicSelfReplicating}_{\mathsf{config}}$, but the replicas do not exhibit more sophisticated properties beyond periodic orbits, and thus ostensibly do not belong to $\mathsf{C}_{2,\,\mathsf{config}}$. At the other end, the Perrier-Sipper-Zahnd loops belong to $\mathsf{UniversalSelfReplicating}_{\mathsf{config}}$. It would therefore be natural to distill a hierarchy of dynamical conditions from these different kinds of self-replicating loops to furnish descriptions of the sets C_i.

We can further set our sights beyond self-replication to ask for necessary conditions on CAs that furnish configurations exhibiting sexual reproduction. This is more challenging since offspring can differ substantially from their parents, making it difficult to quantify family resemblance without presupposing specific organismal structure. We thus pose the following question:

Question H.9. *Are there natural necessary conditions on CA configurations such that they exhibit sexual reproduction?*

Addressing this question, and more broadly identifying dynamical mechanisms for sexual reproduction, would be both mathematically interesting and theoretically important, with consequences for mathematical biology and the philosophy of biology.

Robustness to noise, mutations, and evolution. Our explorations of a theoretical foundation for self-replicating systems have focused on the cleanest setting: dynamics in the absence of environmental noise. A comprehensive theory of self-replication must explain how the mechanisms involved are robust to noise. Moreover, introducing noise raises the possibility of mutations during self-replication, which complicates the identification of 'replicas' since they may no longer be exact copies. This is a more modest version of the problem of abstractly defining sexual reproduction that we mentioned above.

Several noise models for CAs merit consideration. In probabilistic CAs, where the rules themselves are subject to error, robust self-replication would require some form of error correction. However, a crucial distinction must be made: the error correction should be performed by the organism itself, not by the global CA configuration. While it is possible to globally simulate a noiseless CA within a noisy one using error correction schemes, this approach would be inappropriate for studying robust self-replication. For instance, simply embedding von Neumann's universal self-replicator in a globally error-corrected noisy CA would protect the entire configuration rather than demonstrating that the organism itself can handle errors. The organism, which occupies only part of the configuration, should be responsible for its own error correction. Formalizing this distinction between organism-level and global-level error correction would be valuable for understanding robust self-replication.

A different error model fits within our existing paradigm of deterministic CAs. Consider a CA configuration whose dynamics exhibit non-trivial self-replication. We can ask whether this self-replicating behavior remains robust under perturbations of the initial state, both within the organism itself and in its surrounding environment. After all, organisms in nature do not exist in a void; they must contend with external objects and environmental interactions that may interfere with their replication processes.

These considerations of robustness and mutation naturally lead to von Neumann's grander vision for his universal self-replicator construction [1, 114]. He viewed self-replication not as an end in itself, but as an intermediate step toward achieving an artificially evolving system where organisms

would compete to write their own descriptions on the universal constructor's tape, thereby securing their replication. This competition inherently requires the robustness we have been discussing: organisms must not only replicate successfully but do so despite environmental interference and in competition with other replicating entities. The talking-heads model offers a potentially interesting framework for exploring such evolutionary dynamics. If the heads that enable self-replication are both robust to perturbations and limited in number, we might observe emergent competition for these scarce replication resources. Observing the dynamics of such a system could reveal how competition for limited replication machinery drives evolutionary processes in simulated models of life. This motivates the question:

Question H.10. *What are minimal conditions on a CA for evolutionary dynamics to emerge? Specifically, can we characterize CAs where limited resources (such as talking heads) combined with mutation-prone replication lead to competitive exclusion, adaptation, or other hallmarks of evolution?*

This question may initially be most readily addressed empirically, via simulations that may lead to open-ended evolution.

Kinetic models. In his original approach to the theoretical study of self-replication, von Neumann considered 'kinetic models' [1, 114, 115], namely particles in continuous space with interactions, as the dynamics to furnish a self-replicating system. This proved particularly challenging, and his friend Stanislaw Ulam suggested cellular automata as an abstraction of kinetic models, which von Neumann ultimately pursued. We remark that the precise sense in which a CA is an 'abstraction' of a kinetic model is subtle; one might say that a CA faithfully abstracting a kinetic model captures the latter's salient degrees of freedom, but this might make the CA somewhat non-local, with dynamics on a graph rather than $\mathbb{Z}^d$. At the very least, CAs provide natural, readily simulable, visually suggestive models for investigating self-replicating systems.

Today, simulating kinetic models (described by systems of ODEs or PDEs) is much easier than in von Neumann's time. Consequently, exploring self-replicating systems within such models has become more feasible, as one can more readily test ideas. Moreover, techniques like neural ODEs enable backpropagation through ODE dynamics, allowing optimization problems that find configurations with specified properties.

A concrete setting where these ideas have been explored is Lenia [116, 117, 118] and its variants (see e.g. [119, 120, 121]). Lenia is a continuous-space version of a CA that evolves according to a continuum kernel. For various kernel parameters, Lenia exhibits visually compelling dynamics with objects that move, appear to copy themselves, and interact in complex ways. While it may be tempting to regard these interactive objects as artificial models of single-cellular life, we argue that this analogy may not be appropriate for several reasons. First, these objects are approximately the size of the continuum kernel's width, placing them at the smallest length scale of available interactions. Thus, they have the character of fundamental particles in Lenia's dynamics rather than composite structures. Second, in parameter regimes where these objects can 'copy' by splitting, this does not constitute non-trivial self-replication; the dynamical mechanism only produces copies of a single object type and cannot produce copies of different objects. This is analogous to our earlier distinction between a copy machine that can copy any document versus a printer that only reproduces a single loaded file.

This reasoning demonstrates that the underlying logic of our definitions for non-trivial self-replication in CAs generalizes to kinetic models. Indeed, one could provide precise generalizations of our definitions (and many of our results) for various classes of kinetic models. Thus our work

on non-trivial self-replication in CAs yields conceptual insights that organize our understanding of non-trivial self-replication across various model types beyond CAs. To this end, it is natural to ask the following question:

Question H.11. *Can one construct a kernel in Lenia (or one of its natural variants) such that there are configurations with dynamics satisfying a natural analog of the local self-replicating condition?*

We also venture the following conjecture:

Conjecture H.12. *It is possible to build a universal self-replicator within Lenia (or one of its natural variants).*

Addressing the question and establishing (or refuting) the conjecture would be valuable both for understanding Lenia specifically and kinetic models more generally.

Efficient identification of non-trivial self-replication. In studying the dynamics of CAs or kinetic models, we face the practical question of how to observe a simulation and determine whether it exhibits non-trivial self-replication. Ultimately, we would like to identify which CA rules and configurations give rise to non-trivial self-replication empirically, making a practical identification procedure highly valuable.

One immediate difficulty with identifying configurations satisfying the local self-replicating condition is that the condition concerns infinite time-horizon dynamics. It requires an unbounded number of organism copies to be produced and exist simultaneously, necessitating both arbitrarily large times and volumes. The condition examines the outputs of self-replication and verifies that they are dynamically non-trivial and growing in number, but it does not directly address the mechanism of self-replication itself. We do not yet know how to abstract such mechanisms in a way that remains agnostic to the full scope of possible self-reproduction strategies. If such mechanisms could be classified, they might be recognizable in local spatial patches over finite time intervals.[2] This leads to the following question:

Question H.13. *Is it possible to classify or partially characterize dynamical mechanisms for the act of non-trivial self-replication?*

This question relates to our earlier discussion about whether self-replicating organisms require abstracted versions of 'head' and 'tape' degrees of freedom analogous to those in a Turing machine. While it remains unclear whether such degrees of freedom are strictly necessary (and articulating substantively distinct alternatives is challenging), it would be valuable to have an algorithm that identifies head and tape degrees of freedom from observed dynamics.

Recent work [122] has explored using large language models (LLMs) [123, 124] to identify interesting 'life-like' organisms in open-ended simulations. These algorithms attempt to automatically

[2] Related work in the philosophy of biology by Rosen [83] argues that organisms cannot be fully captured by formal computational models. His argument centers on organisms being closed to efficient causation (i.e. every component that produces effects within the organism is itself produced by other components within the organism) and containing internal predictive models of their environment and their own interactions with it. This leads Rosen to argue for the presence of impredicative causal loops that, in a manner analogous to Gödel, cannot be formalized without losing essential properties of life. A consequence would be that identifying life from finite observations of a physical system is impossible. While the soundness of these definitions and arguments has been debated, we note that any simulation of an organism should be intelligible at finite resolution, in finite volume, and over finite time intervals (only idealized organisms exist forever). In such finitary settings, direct appeals to Gödel's incompleteness theorems do not apply. Nevertheless, Rosen's concepts of causal closure and anticipatory systems [82] offer an interesting structural perspective on life that could interface with the ideas in this paper.

identify which simulations, among many with different initial conditions or evolution rules, give rise to life-like forms. This approach leverages our ability to semantically articulate properties of living organisms that we would recognize when we see them, automating this semantic recognition through LLMs that can both process semantically rich instructions and interpret images and videos. It would therefore be useful to develop ordinary language versions of our criteria for non-trivial self-replication that could serve as instructions for LLMs to identify examples in simulations.

Probabilistic bounds for emergence of self-replication. A theory of self-replication should provide probabilistic bounds on whether randomized initial conditions within a fixed CA, or across a probabilistic ensemble of CAs, give rise to living organisms (within a time interval $[0, T]$ for large T). To make this precise, we venture the following conjecture:

Conjecture H.14. *Suppose we have a non-trivial necessary condition for non-trivial self-replication in finite area A. Consider the Game of Life CA with periodic boundary conditions on a square of area A. Then there exists a sufficiently large area A_0 such that for all $A \geq A_0$, there is a $\delta > 0$ such that at least a δ-fraction of all initial conditions exhibit non-trivial self-replication at some point in their dynamics.*

It would be interesting to understand which CA rules maximize the probability of non-trivial self-replication emerging.

A more ambitious goal would be to consider chemical systems present during early Earth's development and establish lower bounds on the probability that they give rise to non-trivial self-replication and ultimately complex life. Addressing this would help us understand how rare or common life is, both on Earth and potentially throughout the universe.

The questions and conjectures above collectively work toward a general theory of self-replication that captures its essential features across different dynamical systems. From understanding the minimal computational requirements to predicting spontaneous emergence, from formalizing evolutionary dynamics to developing practical identification methods, these problems span the theoretical and empirical domains necessary for a comprehensive understanding of self-replication. Progress on this program would fulfill von Neumann's original vision of understanding self-replication as a fundamental phenomenon, independent of its specific physical or biological implementation.

References

[1] John Von Neumann and Arthur W. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press, USA, 1966.

[2] Jarkko Kari. *Cellular automata*. University of Turku, 2022.

[3] Sydney Brenner. Life's code script. *Nature*, 482(7386):461–461, 2012.

[4] Christopher G. Langton. Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):135–144, 1984.

[5] John Byl. Self-reproduction in small cellular automata. *Physica D: Nonlinear Phenomena*, 34(1-2):295–299, 1989.

[6] Hiroki Sayama. Toward the realization of an evolving ecosystem on cellular automata. In *Proceedings of the Fourth International Symposium on Artificial Life and Robotics (AROB 4th'99)*, pages 254–257, 1999.

[7] Moshe Sipper. Fifty years of research on self-replication: An overview. *Artificial life*, 4(3):237–257, 1998.

[8] Martin Gardner. Mathematical games. *Scientific American*, 223(4):120–123, 1970.

[9] Nathaniel Johnston and Dave Greene. *Conway's Game of Life: Mathematics and Construction*. Nathaniel Johnston, 2022.

[10] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning ways for your mathematical plays, Volume 4*. AK Peters/CRC Press, 2004.

[11] Paul Rendell. Turing universality of the game of life. *Collision-based computing*, pages 513–539, 2002.

[12] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[13] Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15, 03 2004.

[14] Turlough Neary and Damien Woods. P-completeness of cellular automaton rule 110. In *International Colloquium on Automata, Languages, and Programming*, pages 132–143. Springer, 2006.

[15] Matthew Cook. A concrete view of rule 110 computation. *arXiv:0906.3248*, 2009.

[16] Nicolas Ollinger. *Universalities in Cellular Automata*, pages 189–229. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] Guillaume Theyssier. *The Mirage of Universality in Cellular Automata*, pages 57–70. Springer International Publishing, 2022.

[18] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[19] Neil D Jones. *Computability and complexity: from a programming perspective*. MIT press, 1997.

[20] Douglas R. Hofstadter. *Gödel, Escher, Bach: an eternal golden braid*. Basic books, 1999.

[21] Jürgen Albert and Karel Culik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.

[22] Kristian Lindgren and Mats G Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4(3):299–318, 1990.

[23] Bruno Durand and Zsuzsanna Róka. The game of life: universality revisited. *Cellular Automata: a Parallel Model*, pages 51–74, 1999.

[24] Jarkko Kari. Reversible cellular automata. In *Developments in Language Theory: 9th International Conference, DLT 2005, Palermo, Italy, July 4-8, 2005. Proceedings 9*, pages 57–68. Springer, 2005.

[25] Gualtiero Piccinini. *Physical computation: A mechanistic account*. Oxford University Press, 2015.

[26] Jacques Mazoyer and Ivan Rapaport. Inducing an order on cellular automata by a grouping operation. In *STACS 98: 15th Annual Symposium on Theoretical Aspects of Computer Science Paris, France, February 25–27, 1998 Proceedings*, pages 116–127. Springer, 2005.

[27] Peter Hertling. Embedding Cellular Automata into Reversible Ones. *Unconventional Models of Computation*, page 243, 1998.

[28] Kenichi Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148(1):157–163, 1995.

[29] Jérôme Olivier Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In Rüdiger Reischuk and Michel Morvan, editors, *STACS 97*, pages 439–450, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[30] Jordan Cotler and Semon Rezchikov. Computational dynamical systems. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 166–202. IEEE, 2024.

[31] J. W. Thatcher. Universality in the von Neumann Cellular Model. In *Essays on Cellular Automata*, pages 132–186. University of Illinois Press, 1970.

[32] Edgar F. Codd. *Cellular Automata*. Academic Press, London, 1968.

[33] Jean-Yves Perrier, Moshe Sipper, and Jacques Zahnd. Toward a viable, self-reproducing universal computer. *Physica D: Nonlinear Phenomena*, 97(4):335–352, 1996.

[34] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012.

[35] Andrew Trevorrow and Tom Rokicki. Golly.

[36] Michael A. Arbib. Simple self-reproducing universal automata. *Information and Control*, 9(2):177–189, 1966.

[37] James A Reggia, Steven L Armentrout, Hui-Hsien Chou, and Yun Peng. Simple systems that exhibit self-directed replication. *Science*, 259(5099):1282–1287, 1993.

[38] Jesús Ibáñez, Daniel Anabitarte, Iker Azpeitia, Oscar Barrera, Arkaitz Barrutieta, Haritz Blanco, and Francisco Echarte. Self-inspection based reproduction in cellular automata. In *European conference on artificial life*, pages 564–576. Springer, 1995.

[39] Kenichi Morita and Katsunobu Imai. Self-reproduction in a reversible cellular space. *Theoretical Computer Science*, 168(2):337–366, 1996.

[40] Gianluca Tempesti. A new self-reproducing cellular automaton capable of construction and computation. In *European conference on artificial life*, pages 555–563. Springer, 1995.

[41] Hui-Hsien Chou and James A Reggia. Problem solving during artificial selection of self-replicating loops. *Physica D: Nonlinear Phenomena*, 115(3-4):293–312, 1998.

[42] James A Reggia, Jason D Lohn, and Hui-Hsien Chou. Self-replicating structures: evolution, emergence, and computation. *Artificial Life*, 4(3):283–302, 1998.

[43] Hiroki Sayama and Chrystopher L. Nehaniv. Self-reproduction and evolution in cellular automata: 25 years after evoloops. *Artificial Life*, 31(1):81–95, 2024.

[44] Paul Bratley and Jean Millo. Computer recreations: Self-reproducing programs. *Software: Practice and Experience*, 2(4):397–400, 1972.

[45] John Burger, Dennis Brill, and Frank Machi. Self-reproducing programs. *Byte*, 5(9):72–74, 1980.

[46] T. S. Ray. An approach to the synthesis of life. *Artificial Life II*, pages 371–408, 1991.

[47] John R Koza. Spontaneous emergence of self-replicating and evolutionarily self-improving computer programs. *Artificial life III*, 17:225–262, 1994.

[48] Blaise Agüera y Arcas, Jyrki Alakuijala, James Evans, Ben Laurie, Alexander Mordvintsev, Eyvind Niklasson, Ettore Randazzo, and Luca Versari. Computational life: How well-formed, self-replicating programs emerge from simple interaction, 2024.

[49] Lionel S Penrose. Mechanics of self-reproduction. *Annals of Human Genetics*, 23(1):59–72, 1958.

[50] Lionel S Penrose and Roger Penrose. A self-reproducing analogue. *Nature*, 179(4571):1183–1183, 1957.

[51] Homer Jacobson. On models of reproduction. *American Scientist*, 46(3):255–284, 1958.

[52] Robert A. Freitas and Ralph C. Merkle. *Kinematic self-replicating machines*. Landes Bioscience, Georgetown, Tex, 2004.

[53] Matthew S Moses and Gregory S Chirikjian. Robotic self-replication. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):1–24, 2020.

[54] Edward F. Moore. Machine models of self-reproduction. In *Proceedings of symposia in applied mathematics*, volume 14, pages 17–33. American Mathematical Society New York, 1962.

[55] Alvy Ray Smith. Simple computation-universal cellular spaces and self-reproduction. In *9th Annual Symposium on Switching and Automata Theory (swat 1968)*, pages 269–277. IEEE, 1968.

[56] Alvy Ray Smith. Simple non-trivial self-reproducing machines. *Artificial life II*, 10:709–725, 1991.

[57] Jason D. Lohn and James A. Reggia. Automatic discovery of self-replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation*, 1(3):165–178, 2002.

[58] CL Nehaniv and Kerstin Dautenhahn. Self-replication and reproduction: Considerations and obstacles for rigorous definitions. In *Third German Workshop on Artificial Life: Abstracting and Synthesizing the Principles of Life*, pages 283–290, 1998.

[59] Bryant Adams and Hod Lipson. A universal framework for self-replication. In *European Conference on Artificial Life*, pages 1–9. Springer, 2003.

[60] Claude E Shannon. A universal turing machine with two internal states. *Automata studies*, 34:157–165, 1956.

[61] Marvin Minsky. A 6-Symbol 7-State Universal Turing Machine, 1960. Reproduced typescript, 8 pages with text diagrams.

[62] Marvin L Minsky. Size and structure of universal turing machines using tag systems. *Journal of Symbolic Logic*, 31(4), 1966.

[63] Shigeru Watanabe. 5-symbol 8-state and 5-symbol 6-state universal turing machines. *Journal of the ACM (JACM)*, 8(4):476–483, 1961.

[64] Shigeru Watanabe. 4-symbol 5-state universal turing machine. *Information Processing Society of Japan Magazine*, 13(9):588–592, 1972.

[65] Raphael M Robinson. Minsky's small universal turing machine. *International Journal of Mathematics*, 2(05):551–562, 1991.

[66] Edwin Roger Banks. Universality in cellular automata. In *11th Annual Symposium on Switching and Automata Theory (swat 1970)*, pages 194–215. IEEE, 1970.

[67] F Noural and R. Sohrab Kashef. A universal four-state cellular computer. *IEEE Transactions on Computers*, 100(8):766–776, 2006.

[68] Alvy Ray Smith III. Simple computation-universal cellular spaces. *Journal of the ACM (JACM)*, 18(3):339–353, 1971.

[69] Turlough Neary and Damien Woods. Small fast universal Turing machines. *Theoretical Computer Science*, 362(1-3):171–195, 2006.

[70] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15(2):213–231, 1977.

[71] Kenichi Morita and Masateru Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *IEICE Transactions (1976-1990)*, 72(6):758–762, 1989.

[72] Jean-Christophe Dubacq. How to simulate turing machines by invertible one-dimensional cellular automata. *International Journal of Foundations of Computer Science*, 6(04):395–402, 1995.

[73] Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354, 1990.

[74] Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(2):199, 1991.

[75] Michael S Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 1995.

[76] Cristopher Moore. Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. In *1st International Conference on Unconventional Models of Computation-UMC*, volume 98, pages 59–71, 1998.

[77] Cristopher Moore. Dynamical recognizers: Real-time language recognition by analog computers. *Theoretical Computer Science*, 201(1-2):99–136, 1998.

[78] Pascal Koiran and Cristopher Moore. Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science*, 210(1):217–223, 1999.

[79] Eugene Asarin and Ahmed Bouajjani. Perturbed Turing machines and hybrid systems. In *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, pages 269–278. IEEE, 2001.

[80] Daniel S Graça, Manuel L Campagnolo, and Jorge Buescu. Robust simulations of Turing machines with analytic maps and flows. In *Conference on Computability in Europe*, pages 169–179. Springer, 2005.

[81] Daniel S Graça, Manuel L Campagnolo, and Jorge Buescu. Computability with polynomial differential equations. *Advances in Applied Mathematics*, 40(3):330–349, 2008.

[82] Robert Rosen. *Anticipatory Systems: Philosophical, Mathematical, and Methodological Foundations*. Springer, 2nd edition, 2012.

[83] Robert Rosen. *Life itself: a comprehensive inquiry into the nature, origin, and fabrication of life*. Columbia University Press, 1991.

[84] Terence Tao. On the universality of potential well dynamics. *arXiv:1707.02389*, 2017.

[85] Robert Cardona, Eva Miranda, Daniel Peralta-Salas, and Francisco Presas. Constructing Turing complete Euler flows in dimension 3. *Proceedings of the National Academy of Sciences*, 118(19):e2026818118, 2021.

[86] Robert Cardona, Eva Miranda, and Daniel Peralta-Salas. Turing universality of the incompressible Euler equations and a conjecture of Moore. *International Mathematics Research Notices*, 2022(22):18092–18109, 2022.

[87] Robert Cardona, Eva Miranda, and Daniel Peralta-Salas. Computability and Beltrami fields in Euclidean space. *Journal de Mathématiques Pures et Appliquées*, 169:50–81, 2023.

[88] Robert Cardona. Hydrodynamic and symbolic models of computation with advice. *Revista Matemática Iberoamericana*, 41(1):313–338, 2024.

[89] Katsunobu Imai and Kenichi Morita. A computation-universal two-dimensional 8-state triangular reversible cellular automaton. *Theoretical Computer Science*, 231(2):181–191, 2000.

[90] Nicolas Ollinger. The quest for small universal cellular automata. In *International Colloquium on Automata, Languages, and Programming*, pages 318–329. Springer, 2002.

[91] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking i: an abstract theory of bulking. *Theoretical Computer Science*, 412(30):3866–3880, 2011.

[92] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking ii: Classifications of cellular automata. *Theoretical Computer Science*, 412(30):3881–3905, 2011.

[93] Barbora Hudcová and Jakub Krásenský. Simulation limitations of affine cellular automata. *Theoretical Computer Science*, 1003:114606, 2024.

[94] Nicolas Ollinger. The intrinsic universality problem of one-dimensional cellular automata. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 632–641. Springer, 2003.

[95] Petr Kurka. *Topological and symbolic dynamics*. Société mathématique de France Paris, France, 2003.

[96] Daniel Gonçalves and Marcelo Sobottka. Continuous shift commuting maps between ultragraph shift spaces. *Discrete and Continuous Dynamical Systems*, 39(2):1033–1048, 2019.

[97] Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, 2021.

[98] Jacques Mazoyer and Ivan Rapaport. Additive cellular automata over $\mathbb{Z}_p$ and the bottom of $(ca, \leq)$. In *Mathematical Foundations of Computer Science 1998: 23rd International Symposium, MFCS'98 Brno, Czech Republic, August 24–28, 1998 Proceedings 23*, pages 834–843. Springer, 1998.

[99] Nicolas Ollinger. *Automates cellulaires: structures*. PhD thesis, Ecole Normale Supérieure de Lyon, 2002.

[100] Guillaume Theyssier. *Automates cellulaires: un modele de complexités*. PhD thesis, Ecole Normale Supérieure de Lyon, 2005.

[101] Barbora Hudcová. *Complexity and Computational Capacity of Discrete Dynamical Systems*. PhD thesis, Charles University, 2024.

[102] Jean-Charles Delvenne, Petr Kůrka, and Vincent Blondel. Decidability and universality in symbolic dynamical systems. *Fundamenta Informaticae*, 74(4):463–490, 2006.

[103] Klaus Sutner. Universality and cellular automata. In *International Conference on Machines, Computations, and Universality*, pages 50–59. Springer, 2004.

[104] Jarkko Kari. Reversibility of 2d cellular automata is undecidable. *Physica D: Nonlinear Phenomena*, 45(1-3):379–385, 1990.

[105] Jérôme Olivier Durand-Lose. Reversible space–time simulation of cellular automata. *Theoretical computer science*, 246(1-2):117–129, 2000.

[106] William R Buckley. Signal crossing solutions in von neumann self-replicating cellular automata. In *Automata*, pages 453–503, 2008.

[107] Jérôme O Durand-Lose. Reversible space-time simulation of cellular automata. *Theoretical Computer Science*, 246(1-2):117–129, 2000.

[108] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, IL, 2002.

[109] Jarkko Kari and Siamak Taati. Statistical mechanics of surjective cellular automata. *Journal of Statistical Physics*, 160(5):1198–1243, 2015.

[110] Silvio Capobianco, Pierre Guillon, and Jarkko Kari. Surjective cellular automata far from the Garden of Eden. *Discrete Mathematics & Theoretical Computer Science*, 15(Automata, Logic and Semantics), 2013.

[111] Cristian S Calude, Peter H Hertling, Helmut Jürgensen, and Klaus Weihrauch. Randomness on full shift spaces. *Chaos, Solitons & Fractals*, 12(3):491–503, 2001.

[112] Norman Margolus. Crystalline computation. In *Feynman and Computation*, pages 267–305. CRC Press, 2018.

[113] Mikhael Gromov. *Geometric Group Theory: Volume 2*. Cambridge University Press, 1993.

[114] John Von Neumann and AH Taub. *The general and logical theory of automata*, volume 5. MIT Press Cambridge, 1963.

[115] Arthur Walter Burks. *Essays on cellular automata*. University of Illinois Press, 1970.

[116] Bert Wang-Chak Chan. Lenia-biology of artificial life. *arXiv:1812.05433*, 2018.

[117] Bert Wang-Chak Chan. Lenia and expanded universe. In *Artificial Life Conference Proceedings 32*, pages 221–229. MIT Press, 2020.

[118] Bert Wang-Chak Chan. Towards large-scale simulations of open-ended evolution in continuous cellular automata. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 127–130, 2023.

[119] Erwan Plantec, Gautier Hamon, Mayalen Etcheverry, Pierre-Yves Oudeyer, Clément Moulin-Frier, and Bert Wang-Chak Chan. Flow-Lenia: Towards open-ended evolution in cellular automata through mass conservation and parameter localization. In *Artificial Life Conference Proceedings 35*, page 131. MIT Press, 2023.

[120] Alexander Mordvintsev, Eric Niklasson, and Ettore Randazzo. Particle Lenia and the energy-based formulation, 2022. Google Research Blog, Self-Organising Systems.

[121] Vassilis Papadopoulos and Etienne Guichard. MaCE: General Mass Conserving Dynamics for Cellular Automata. *arXiv:2507.12306*, 2025.

[122] Akarsh Kumar, Chris Lu, Louis Kirsch, Yujin Tang, Kenneth O Stanley, Phillip Isola, and David Ha. Automating the search for artificial life with foundation models. *arXiv:2412.17799*, 2024.

[123] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[124] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.