Physica D

Research Article

Introducing Encoder–Decoder Relation between Cellular Automata to Uncover their Computational Structure

Barbora Hudcová [1,2,3], Tomáš Mikolov [2] and Stefano Nichele [3,4]

[1]Charles University, Prague, Czech Republic

[2]Czech Institute of Informatics, Robotics and Cybernetics, CTU, Prague, Czech Republic

[3]Oslo Metropolitan University, Oslo, Norway

[4]Østfold University College, Halden, Norway

Correspondence should be addressed to Barbora Hudcová: bara.hudcova@gmail.com, address: Petržílkova 2489, Prague 5, 158 00, Czech Republic

Abstract
Cellular automata are fascinating models of parallel computation, yet it remains a challenge to program them efficiently to solve complex tasks. In this paper, we propose a novel method of analyzing their computational capacity. It is based on measuring the richness of the automaton's dynamics by assessing how many other automata it can simulate. Using this method, we can identify automata that can perform many computations efficiently. Moreover, there is no need to choose an arbitrary set of computational tasks as they are self-defined by the cellular automata space itself. The novelty of our method is based on a new definition of automata simulation that generalizes previous approaches. Thus, we obtain a rich structure of relations that we demonstrate on the class of elementary cellular automata. We show that the number of elementary automata with unique dynamics can be reduced from 88 to 52. We further show that some automata, such as 14, 43, and 142, have in fact equivalent computational capacity. We believe that using similar approaches, the number of unique automata can be dramatically reduced in the future.

Keywords: computation in parallel systems, cellular automata simulation, computational equivalence

1 Introduction

Cellular automata (CAs) are a promising model of efficient computation due to their massively parallel nature and self-organization capability ([30], [1], [25]), yet it remains a challenge to find an effective way of programming them to solve complex tasks in practice.

The aim of this paper is to bring more insight into the computational structure of such parallel systems. In the classical context of computation defined by Turing machines, the most powerful machine is the one that can simulate the computation of any other Turing machine. Similarly, we can study the computational capacity of a cellular automaton by measuring how many other automata it can simulate. Thus, we could identify systems that can perform many computations efficiently without having to arbitrarily choose a task to evaluate their performance on. The tasks are defined from within, by the space of the systems themselves. The general paradigm of relative simulation is described as follows.

We say that a system S_1 can be simulated by S_2 if we can encode any input of S_1 to obtain an input of S_2, let S_2 compute the output, and decode this output to obtain the resulting computation of S_1. This concept of *relative simulation* is illustrated in Figure 1.

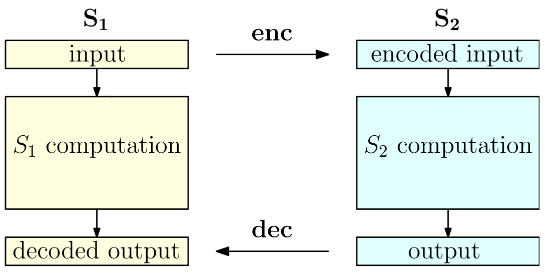


Figure 1: Scheme of computational system S_1 being simulated by S_2.

In such a case, we conclude that S_2 is computationally stronger than (or equal to) S_1 as any input–output mapping realized by S_1 can also be obtained by using S_2; we write $S_1 \leq S_2$. If it also holds that $S_2 \leq S_1$, we say the two systems are computationally equivalent. The concrete definition of relative simulation for a particular class of systems then depends on how exactly the encoding and decoding mappings are defined.

In this paper, we introduce the notion of relative simulation for cellular automata. In contrast to the relative simulation of Turing machines, it will follow that our definition takes into account the efficiency of the computation as well; we will have a guarantee that if ca_1 is simulated by ca_2, then there exists a constant k such that whenever ca_1 needs t time steps to compute a task, ca_2 needs at most $k \cdot t$ time steps to perform the same task.

Further, we will discuss that our notion generalizes previous approaches and allows us to find a rich structure of relations between cellular automata. We showcase the results on the class of elementary cellular automata (ECAs). We will argue that if $\mathrm{ca}_1 \leq \mathrm{ca}_2$ then the dynamics of ca_1 are contained in the dynamics of ca_2. Thus, we show that the number of ECAs with unique dynamics can be reduced from 88 to 52. Such space reductions can be particularly useful in large spaces of CAs where we wish to apply search methods in order to find automata with complex behavior or capability to solve challenging tasks efficiently.

Moreover, we show that certain ECAs previously considered to be unique have in fact equivalent computational capacity; for example, we show that the ECAs 14, 43, and 142 are computationally equivalent. We believe that with similar methods, the future results could show that the number of ECAs with unique computational capacities could be reduced dramatically.

2 Cellular Automata

Cellular automata (CAs) are discrete space and time dynamical systems. They consist of a d-dimensional grid of identical finite state automata, each updated locally and synchronously in discrete time steps. A formal, more detailed definition can be found in [14].

CAs were first introduced as models of self-replicating structures [22, 15] and subsequently became widely studied for their complex dynamics emerging from the iterations of very simple local rules [9, 10, 7]. As such, they were examined as potential models of artificial evolution [16, 27] and their complexity has been analyzed from diverse angles [6, 26, 31, 32]. The next section introduces some basic notions connected to cellular automata.

2.1 Notation

For simplicity of notation, we will study one-dimensional cellular automata (1D CAs) in this paper. We call the set of integers $\mathbb{Z}$ the *one-dimensional grid* and we call its elements *cells*. For a finite set S, we define an S-configuration as a bi-infinite sequence $c = \ldots c_{-1}c_0c_1 \ldots \in S^{\mathbb{Z}}$ of elements from S, indexed by $\mathbb{Z}$. Each 1D CA is characterized by a tuple (S, f) where S is a finite set of *states* and $f : S^{2r+1} \to S$ is the *local transition rule with radius* r. The *global rule* of the CA (S, f) *operating on the infinite grid* is a mapping $F : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ defined as:

$$F(c)_i = f(c_{i-r}, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_{i+r}) \qquad (1)$$

for each $i \in \mathbb{Z}$. For practical purposes, when observing the CA iterations, we often consider the grid to be finite with a periodic boundary condition. In such a case, we compute the indices in (1) modulo the size of the grid. For a CA (S, f) operating on a periodic grid of size n with global rule F we define the *trajectory of a configuration* $u \in S^n$ as the infinite sequence $(u, F(u), F^2(u), \ldots)$. We define a *space-time diagram* of the CA to be a matrix whose rows are exactly $u, F(u), F^2(u), \ldots, F^t(u)$ for some configuration $u \in S^n$ and time step $t \in \mathbb{N}$.

2.2 Elementary CAs

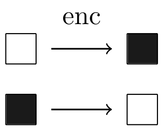
1D cellular automata with states $\mathbf{2} = \{0, 1\}$ and a local rule with radius 1 are called elementary cellular automata (ECAs). Each local rule $f : \mathbf{2}^3 \to \mathbf{2}$ is uniquely described by its *Wolfram number* given by:

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

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 [5].

2.3 CA Simulation: First Simple Example

Let us consider ECAs 110 and 137. One is obtained by the other by exchanging the role of the 0 (white color) and 1 state (black color). Then, any space-time diagram of ECA 110 can be encoded into a space-time diagram of ECA 137 using the simple encoding:



We simply apply the encoding cell by cell to ECA 110 space-time diagram to obtain a space-time diagram of ECA 137, as illustrated in Figure 2.

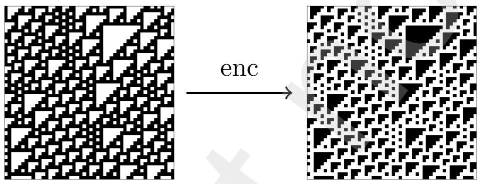


Figure 2: ECA 110 space-time diagram (left) encoded into an ECA 137 space-time diagram (right).

Further, we can observe that using the same encoding, we can also map every space-time diagram of ECA 137 into a space-time diagram of ECA 110. In fact, the encoding yields a bijection between the space-time diagrams of the two ECAs implying that the ECAs have equivalent dynamics.

The interesting question is whether we can introduce more general relations between the diagrams to enrich the structure of relationships between two ECAs. We introduce such generalized relations in the next section.

3 CA Enc–Dec Simulation

In this section, we define the notion of CA simulation for 1D cellular automata. We will first introduce the notion of larger scale dynamics of a CA: informally, we group together blocks of cells and view them as new, richer states upon which the CA operates. Then, the notion of CA simulation can be explained as follows: ca_1 can be simulated by ca_2 if the dynamics of ca_1 are contained in the larger scale dynamics of ca_2.

3.1 CA Enc–Dec Simulation: Introduction

3.1.1 Larger Scale of CA Dynamics

Let (T, g) be a CA with global rule $G : T^{\mathbb{Z}} \to T^{\mathbb{Z}}$ and let $k \in \mathbb{N}$. We can group each k consecutive cells together into a “larger" cell called the *k-block* and view the CA as operating on this richer state space T^k. To give the CA “more time to process this richer information" we consider the operation to consist of k iterations of the global rule G. We illustrate this concept in Figure 3 and define the function operating on this larger scale as G_k; $G_k : (T^k)^{\mathbb{Z}} \to (T^k)^{\mathbb{Z}}$.

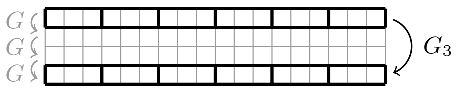


Figure 3: Larger scale behavior of a CA with global rule G, obtained by grouping 3 cells.

Formally, there is a natural bijection $\varphi : (T^k)^{\mathbb{Z}} \to T^{\mathbb{Z}}$ given by $\varphi(c)_{ki}\varphi(c)_{ki+1} \ldots \varphi(c)_{ki+k-1} = c_i \in T^k$ for each $c \in (T^k)^{\mathbb{Z}}$ and $i \in \mathbb{Z}$. Then, we define the new global map $G_k : (T^k)^{\mathbb{Z}} \to (T^k)^{\mathbb{Z}}$ as: $G_k = \varphi^{-1} \circ G^k \circ \varphi$. More details and algebraic motivation behind this notion can be found in the Appendix, Section CA Large scale dynamics: Algebraic motivation.

3.1.2 Defining CA Enc–Dec Simulation

Let S and T be finite sets and $e : S \to T$ an arbitrary mapping. By $\overline{e}$ we will denote the extended mapping $\overline{e} : S^{\mathbb{Z}} \to T^{\mathbb{Z}}$, which simply puts $\overline{e}(c)_i = e(c_i)$ for every $c \in S^{\mathbb{Z}}$ and $i \in \mathbb{Z}$.

Definition 1 (CA enc–dec simulation)**.**
Let $\mathrm{ca}_1 = (S, f)$ and $\mathrm{ca}_2 = (T, g)$ be two 1D cellular automata with global rules F and G respectively. We say that ca_1 *can be enc–dec simulated by* ca_2 *with an*

encoding of size $k \in \mathbb{N}$ if there exists an injective encoding $\mathrm{enc} : S \to T^k$ and a decoding $\mathrm{dec} : T^k \to S$ such that for all initial configurations $c \in S^{\mathbb{Z}}$ and for all time steps $t \in \{0, 1, 2, \ldots\}$ it holds that

$$F^t(c) = \overline{\mathrm{dec}}\left(G_k^t\left(\overline{\mathrm{enc}}\left(c\right)\right)\right).$$

In such a case, we write $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ and say that enc and dec *witness* this relation. We write $\mathrm{ca}_1 \leq \mathrm{ca}_2$ if there exists some k for which $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ and say that ca_2 can *enc–dec simulate* ca_1. The notion of CA simulation is illustrated in Figure 4. This notion is further illustrated in Example of Enc–Dec Simulation.

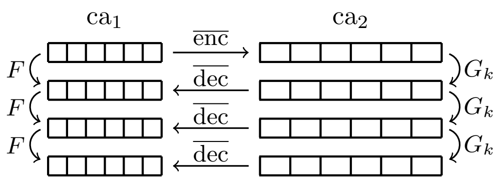

ca$_1$
ca$_2$
$\overline{\mathrm{enc}}$
$\overline{\mathrm{dec}}$
$\overline{\mathrm{dec}}$
$\overline{\mathrm{dec}}$
F
F
F
G_k
G_k
G_k

Figure 4: Diagram of ca_1 enc–dec simulated by ca_2.

For conciseness, we will refer to the enc–dec simulation simply as simulation in the following text. Trivially, for each cellular automaton ca it holds that $\mathrm{ca} \leq_1 \mathrm{ca}$. If $\mathrm{ca} \leq_k \mathrm{ca}$ for some $k \geq 2$, we say that ca can simulate itself non-trivially and that it is *self-similar*. If it holds that both $\mathrm{ca}_1 \leq \mathrm{ca}_2$ and $\mathrm{ca}_2 \leq \mathrm{ca}_1$ we say that ca_1 and ca_2 are *computationally equivalent*.

3.2 Basic Properties of CA Simulation

Let us fix the notation from Definition 1 and suppose we have an initial configuration $c \in S^{\mathbb{Z}}$ of ca_1, and we would like to know what the configuration looks like after t time steps. If ca_1 is simulated by ca_2, we can simply encode c using the local encoding, iterate ca_2 on the encoded configuration for $k \cdot t$ time steps and decode the outcome. Thus, we can compute the iterations of ca_1 just by using the automaton ca_2. In such a case, it is natural to say that ca_2 is computationally stronger than (or equal to) ca_1. As a special case, let us consider ca_1 that is proven to be Turing complete by showing it emulates a Universal Turing Machine in its space-time diagrams (such as in [5]). Then, if ca_1 is simulated by ca_2, this implies that ca_2 is Turing complete as well.

We note that a CA (S, f) operating on a cyclic grid with a periodic boundary condition is equivalent to a CA operating on an infinite grid with configurations consisting of periodically repeating patterns. Therefore, if ca_1 is simulated by ca_2 on an infinite grid, it also is the case for the periodic grid of arbitrary size.

We note that in Definition 1 the encoding and decoding must "work" also at time step $t = 0$. This implies that $\mathrm{dec} \circ \mathrm{enc} = \mathrm{id}$. Thus, $\mathrm{dec} : T^k \to S$ must be onto and must be an extension of enc^{-1}. Due to this, as the decoder cannot be a constant 0 mapping, even simulating ECA 0 (the constant 0 mapping), is non-trivial.

In the Appendix, we give proof of the important fact that the simulation relation $\leq$ is reflexive and transitive and, therefore, forms a preorder. This will allow us, among other things, to represent the resulting hierarchy of the $\leq$ relation for ECAs in a compact way without having to plot all the edges explicitly.

3.3 Example of Enc–Dec Simulation

We show that ECA 14 can simulate ECA 43 with an encoding of size 3.

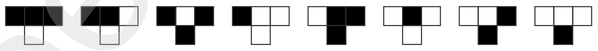


Figure 5: ECA 43 rule table.

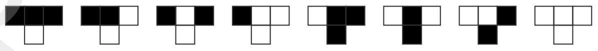


Figure 6: ECA 14 rule table.

The witnessing encoding and decoding are defined as follows:

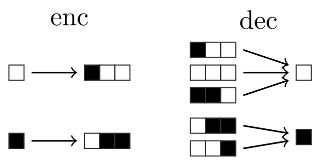


Figure 7: Encoding and decoding of size 3 witnessing that ECA 14 can simulate ECA 43.

In Figure 7, the presented decoder is only a partial mapping. From the Verification Algorithm described in the Appendix we obtained the result that no other 3-blocks occur in the large scale dynamics of ECA 14 iterated on encoded initial configurations. Therefore, the decoder can be assigned arbitrary values on such blocks.

Let F and G denote the global rules of ECA 43 and ECA 14, respectively, both operating on a periodic grid.

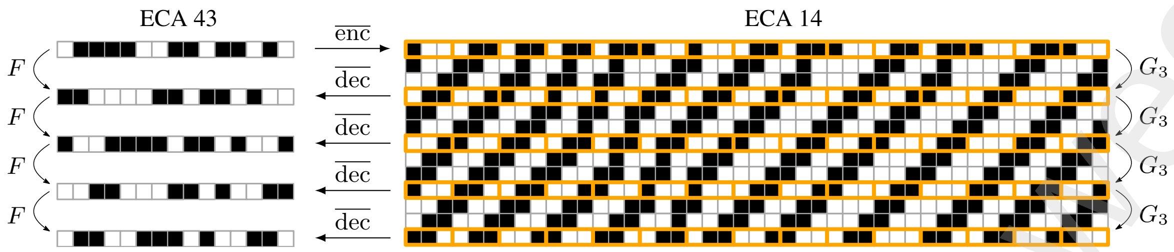


Figure 8: Diagram shows ECA 14 simulating ECA 43 on a particular periodic grid configuration $c \in \mathbf{2}^{15}$.

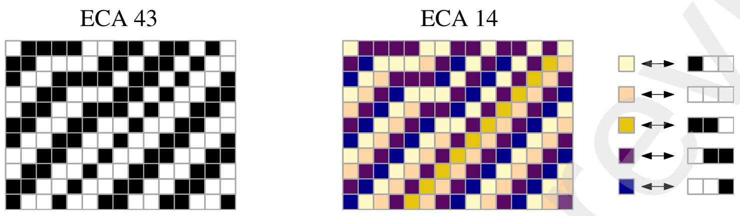


Figure 9: Compressed space-time diagram of ECA 14 is showed on the right to make it better visible that the corresponding space-time diagram of ECA 43 is “contained" in it.

In Figure 8, we show ECA 14 simulating ECA 43 on a particular periodic grid configuration $c \in \mathbf{2}^{15}$.

In the space-time diagram of ECA 14, only every third iteration of the rule is relevant for the simulation relation. Such rows are highlighted in orange in Figure 8. In Figure 9, we plot the larger scale dynamics of ECA 14, from which it is apparent that the space-time diagram of ECA 14 contains the one of ECA 43. The larger scale diagram is obtained by plotting only every third row of the space-time diagram of ECA 14 and representing the 3-blocks of cells with new colors.

Since the dynamics of both ECAs have stabilized into a periodic attractor, it is easy to verify that the decoding works correctly for an arbitrary time step, not just for the ones depicted in Figure 8. However, verifying that the encoding and decoding given in Figure 7 indeed work for an arbitrary (infinite) configuration is not straightforward at all. In fact, it is a very interesting open problem whether there exists an algorithm deciding whether a given enc–dec pair of size k witnesses the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$.

In Appendix, Section Verification Algorithm, we present a criterion that can be efficiently verified and we prove that if a ca_1, ca_2, and a enc–dec pair of size k satisfy the criterion, we have a guarantee that the enc–dec pair witnesses the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$.

3.4 Previous Work

We discuss previously studied notions of CA simulation and analyze how they relate to the enc–dec simulation.

CA coarse-graining Inspired by the notion of coarse-graining of dynamical systems from physics, Israeli and Goldenfeld studied its analog for cellular automata in [13]. Informally, a ca_2 can be coarse-grained into ca_1 if ca_2 yields the dynamics of ca_1 after “abstracting away" some fine-grained details in the dynamics of ca_2.

Formally, let $\mathrm{ca}_1 = (S, f)$ and $\mathrm{ca}_2 = (T, g)$ be CAs with global rules F and G respectively. We say that ca_2 *can be coarse-grained into* ca_1 *with a supercell of size* k if there exists a surjective decoding $\mathrm{dec} : T^k \to S$ such that for every initial configuration $c \in (T^k)^{\mathbb{Z}}$ of ca_2 and for every time step $t \in \{0, 1, 2, \dots\}$ it holds that $F^t(\overline{\mathrm{dec}}(c)) = \overline{\mathrm{dec}}(G_k^t(c))$, as shown in Figure 10.

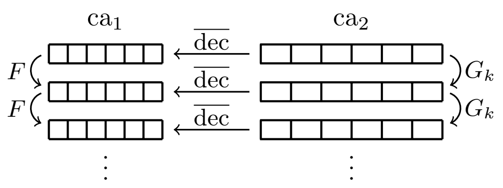


Figure 10: Diagram of ca_2 coarse-grained into ca_1.

From Figure 10, it is easy to see that if ca_2 can be coarse-grained into ca_1 with a supercell of size k then

$ca_1 \leq_k ca_2$. Indeed, since $\mathrm{dec} : T^k \to S$ is surjective, for every state $s \in S$ we can choose $\mathrm{enc}(s)$ to be any element of $\mathrm{dec}^{-1}(s) \neq \emptyset$. Then, it is easy to see that $\mathrm{enc} : S \to T^k$ and $\mathrm{dec} : T^k \to S$ witness $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. Thus, the CA simulation we introduce in this paper generalizes CA coarse-graining. This generalization makes a difference in the resulting relations. In [13], Israeli and Goldenfeld produce a hierarchy of ECAs based on their coarse-grainings. For example, after examining the encodings of sizes 2, 3, and 4, they found no relation between ECA 14 and 43, whereas with the generalized notion, we could find that $43 \leq_3 14$.

CA subautomaton In [20], Mazoyer and Rapaport introduce the notion of a CA *subautomaton*. Based on this, they introduce a CA preorder and study its interesting theoretical properties. This notion is further generalized to allow for geometrical transformations of the space-time diagrams in [8].

Based on the notion of subautomaton, we introduced the notion of *CA emulation* in our previous paper [11] where we studied the resulting hierarchy for ECAs. Informally, we say that ca_1 can be emulated by ca_2 if each space-time diagram of ca_1 can be, at each time step, embedded into a larger scale space-time diagram of ca_2, see Figure 11.

Formally, let $\mathrm{ca}_1 = (S, f)$ and $\mathrm{ca}_2 = (T, g)$ be CAs with global rules F and G respectively. We say that ca_1 can be *emulated* by ca_2 *with a supercell of size* k if there exists an injective encoding $\mathrm{enc} : S \to T^k$ such that for every initial configuration $c \in S^{\mathbb{Z}}$ of ca_1 and for every time step $t \in \{0, 1, 2, \ldots\}$ it holds that $\overline{\mathrm{enc}}(F^t(c)) = G_k^t(\overline{\mathrm{enc}}(c))$, as illustrated in Figure 11.

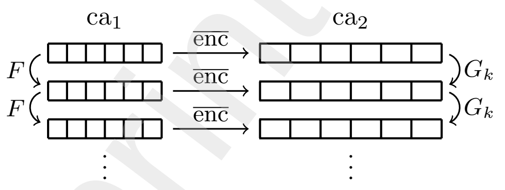

ca_1
ca_2
$\overline{\mathrm{enc}}$
$\overline{\mathrm{enc}}$
$\overline{\mathrm{enc}}$
F
F
G_k
G_k

Figure 11: Diagram of ca_1 emulated by ca_2.

Again, it is straightforward to see that if we put $\mathrm{dec} := \mathrm{enc}^{-1}$ (and define the values of dec outside of the range of enc arbitrarily) then enc and dec witness that $ca_1 \leq_k \mathrm{ca}_2$. Again, the fact that the notion of CA simulation is more general is demonstrated by the resulting hierarchies. Using the same example, in [11], we were not able to find any relation between ECAs 14 and 43 when examining supercells of sizes 2 to 11, whereas with CA enc–dec simulation, we obtain that $43 \leq_3 14$. In the next section, we describe the process of searching for encoding–decoding pairs witnessing a given CA simulation relation.

4 Search Methods

Our goal is to discover a rich structure of relationships between cellular automata based on the simulation relation. In order to do that, it is crucial to find an efficient search method for the encoder–decoder pairs witnessing a relation. This generally seems to be a hard discrete optimization problem. In this chapter, we describe the approach that helped us find 548 relations between ECAs. We, however, believe there is a potential for further improvement in efficiency. For simplicity of notation, we restrict our study to ECAs in this section, though the results could be generalized to any 1D CAs in a straightforward way.

We aim to find an efficient method of solving the following computational task:

Algorithm 1: Simulation Search Algorithm

```
Input  : $\mathrm{ca}_1 = (\mathbf{2}, f)$, $\mathrm{ca}_2 = (\mathbf{2}, g)$, supercell size $k$
Output: a witnessing enc and dec if $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$;
         "unsuccessful" message otherwise
```

To verify that a particular enc–dec pair witnesses the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ we would, in principle, have to check that the diagram in Figure 4 commutes for every time step and every infinite initial configuration. That gives us infinite conditions to check. Therefore, we separate the problem into two parts:

1. First, we describe an algorithm that finds "good candidates" for an enc–dec pair. We do this by designing a training and testing set and searching for an encoder–decoder pair with a perfect score on such sets.
2. Subsequently, we present a criterion that can be efficiently verified for a candidate enc–dec pair. If the criterion is met, it guarantees that the enc–dec pair indeed witnesses the given simulation relation.

In this section, we focus on part 1. described above. The criterion in part 2. is presented in detail in Appendix, Section Verification Algorithm.

In principle, once we fix the encoding size k, there are only finitely many mappings $\mathrm{enc} : \mathbf{2} \to \mathbf{2}^k$ and $\mathrm{dec} : \mathbf{2}^k \to \mathbf{2}$. However, it is not feasible to search through all of them as soon as $k \geq 13$. We now focus on describing our attempt at making this search efficient. In the rest of this section, we fix a particular $k \in \mathbb{N}$, and $\mathrm{ca}_1 = (\mathbf{2}, f)$, $\mathrm{ca}_2 = (\mathbf{2}, g)$ two ECAs with global rules F and G respectively.

Assigning a score to an enc–dec **pair** Suppose we have an encoding $\mathrm{enc} : \mathbf{2} \to \mathbf{2}^k$ and a decoding $\mathrm{dec} : \mathbf{2}^k \to \mathbf{2}$. We want to evaluate how close is the enc–dec pair to perfectly witnessing the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$.

Let us pick a finite testing initial configuration $c \in \mathbf{2}^l$ on a periodic grid for some $l \in \mathbb{N}$ and a time step $t \in \mathbb{N}$. We will measure how well the enc–dec pair works on c at time step t. Let us denote

$$c^t = F^t(c) \in \mathbf{2}^l,$$
$$d^t = G_k^t(\overline{\mathrm{enc}}(c)) \in (\mathbf{2}^k)^l.$$

We compute the score

$$s_{c,t} = |\{i \,|\, dec(d_i^t) = c_i^t\}|, \;\; 0 \leq s_{c,t} \leq l$$

which counts the number of cells that get decoded correctly at time step t.

To build the training set, we generate a set of pairs $\mathrm{Tr} = \{(c_1, t_1), \ldots, (c_n, t_n)\}$ where each $c_i \in 2^{l_i}$ is a testing initial configuration and $t_i \in \mathbb{N}$ is a time step at which we evaluate the score. Given a particular enc–dec pair, for each $i \in \{1, \ldots, n\}$ we can compute the score $s_i = s_{c_i,t_i}$ of correctly decoded k-blocks on the initial configuration c_i at time step t_i. We assign the pair enc–dec a final score $S^{\mathrm{Tr}}_{\mathrm{enc,dec}}$ on the training set Tr as follows:

$$S^{\mathrm{Tr}}_{\mathrm{enc,dec}} = \frac{s_1 + \ldots + s_n}{l_1 + \ldots l_n}.$$

If $S^{\mathrm{Tr}}_{\mathrm{enc,dec}} = 1$, then all the cells at all the time steps were decoded correctly, and we conclude that enc–dec could be a good candidate pair for witnessing $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. We subsequently test whether the enc–dec pair generalizes well and reaches a perfect score on a testing set $\mathrm{Te} = \{(c_1', t_1'), \ldots, (c_m', t_m')\}$. If yes, we proceed to step 2 and try to verify whether the enc–dec indeed witnesses $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$.

Searching for a good candidate decoder Let us fix an encoder $\mathrm{enc} : \mathbf{2} \to \mathbf{2}^k$. Then, finding a decoder that maximizes the score $S^{\mathrm{Tr}}_{\mathrm{enc,dec}}$ on a training set is straightforward, and we describe the process below.

Let $c \in \mathbf{2}^l$ be an initial configuration for ca_1, and $t \in \mathbb{N}$ a time step. Then, given enc, we can compute $c^t = F^t(c)$, and $d^t = G_k^t(\overline{\mathrm{enc}}(c))$ solely without the knowledge of dec. Let $0 \leq i < l$. We know that if the dec should decode the i-th cell correctly, it has to map d_i^t to c_i^t. Thus, we simply put $\mathrm{dec}(d_i^t) := c_i^t$. The only complication occurs when for some i we get the requirement that $\mathrm{dec}(d_i^t) = 0$ and for some $j \neq i$ such that $d_j^t = d_i^t$ we get the "contradicting" requirement $\mathrm{dec}(d_j^t) = 1$. In that case, we choose the decoder's value on d_i^t based on the more frequented requirement to maximize the score. During training, we compute the total frequency of all the requirements on all the configurations from the training set and determine the value of the decoder to maximize the score $S^{\mathrm{Tr}}_{\mathrm{enc,dec}}$.

We note that there might be blocks $d \in T^k$ which never occur in the space-time diagrams of ca_2 during training. For such d, the value of the decoder is not determined by the previously described method, and we build just a partial mapping $\mathrm{dec} : \mathbf{2}^k \dashrightarrow \mathbf{2}$. This is, however, advantageous as the description of the decoder is more compact. If we encounter a block outside of the decoder's domain during testing, we simply conclude the *enc–dec* cannot have a perfect score on the test set and is therefore not a good candidate pair.

We conclude that given an $\mathrm{enc} : \mathbf{2} \to \mathbf{2}^k$, it is straightforward to compute the score

$$\mathbb{S}^{\mathrm{Tr}}_{\mathrm{enc}} := \max\{S^{\mathrm{Tr}}_{\mathrm{enc,dec}} \,|\, \mathrm{dec} : \mathbf{2}^k \to \mathbf{2}\}.$$

Searching for a good candidate encoder The main challenge lies in efficiently searching for an encoder maximizing the score $\mathbb{S}^{\mathrm{Tr}}_{\mathrm{enc}}$ on the training set. Each encoder $enc : \mathbf{2} \to \mathbf{2}^k$ can be identified with a sequence $\mathbf{2}^{2k}$ obtained by concatenating $enc(0)$ and $enc(1)$. There are multiple methods for solving such a discrete optimization problem, ranging from simple random search to elaborate evolutionary algorithm methods. In our case, we need a fast algorithm, as we need to evaluate it many times: for every pair of ECAs and multiple encoder sizes (ranging from 2 to 24). Thus, the algorithm has to be executed hundreds of thousands times. After comparing various approaches, the one able to find good candidates in a short time is the simple hill climbing algorithm described in 2.

If the "unsuccessful" message is returned before the time limit of the computation is reached, a new encoder

Algorithm 2: Hill climbing algorithm for encoder–decoder search.

```
Input  : $\mathrm{ca}_1 = (\mathbf{2}, f)$, $\mathrm{ca}_2 = (\mathbf{2}, g)$, encoder size $k$,
         training set Tr, testing set Te, time bound for the
         algorithm
Output: a pair enc, dec with testing score $S^{\mathrm{Te}}_{\mathrm{enc,dec}} = 1$;
         or "unsuccessful" message
1  randomly generate $\mathrm{enc} = \mathrm{enc}(0)\mathrm{enc}(1) \in \mathbf{2}^{2k}$
2  while computation time $\leq$ time bound do
3      find the optimal dec to maximize $S^{\mathrm{Tr}}_{\mathrm{enc,dec}}$
4      if $S^{\mathrm{Tr}}_{\mathrm{enc,dec}} = 1$ then
5          compute testing score $S^{\mathrm{Te}}_{\mathrm{enc,dec}}$
6          if $S^{\mathrm{Te}}_{\mathrm{enc,dec}} = 1$ then return enc, dec
7          else return "unsuccessful" [the enc, dec pair does
             not generalize well on testing set]
8      end
9      else
10         for $i \in \{0, \dots, 2k-1\}$ do
11             $\mathrm{enc}_i$ = enc mutated at $i$-th position
12             find the optimal $\mathrm{dec}_i$ to maximize $S^{\mathrm{Tr}}_{\mathrm{enc}_i,\mathrm{dec}_i}$
13         end
14         pick the $i$ with the highest training score $S^{\mathrm{Tr}}_{\mathrm{enc}_i,\mathrm{dec}_i}$
15         if $S^{\mathrm{Tr}}_{\mathrm{enc,dec}} < S^{\mathrm{Tr}}_{\mathrm{enc}_i,\mathrm{dec}_i}$ then enc = $\mathrm{enc}_i$
16         else return "unsuccessful" [the local minimum has
             been reached but the perfect training score was not
             obtained]
17     end
18 end
```

is randomly generated, and the process is repeated until we run out of time. If the algorithm returns an enc, dec pair, we have a good candidate for mappings witnessing the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. Subsequently, we verify whether this pair indeed witnesses the relation. For that, we present a criterion that can be verified efficiently. Due to its rather technical nature, we describe it in the Appendix, Section Verification Algorithm.

We note that for all the relations between ECAs we present in the next section, we checked this criterion, and it was met by the encoder–decoder pair. Therefore, for the results presented, we have rigorous proof that the simulation relations we present work for all initial configurations and all time steps.

The implementation of Algorithm 2 can be found in our GitHub Repository. [1]

[1]https://github.com/barahudcova/eca_enc_dec_reductions

5 Results

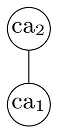
In this section, we present the simulation relations we found between ECAs. First, we clarify how we present the results.

5.1 Representation of the Results

Classically, the number of ECAs is reduced in the following way. We say that two ECAs are equivalent if one's local rule is obtained from the other by either changing the role of its left and right input bit, the role of 0 and 1 state, or both. This produces equivalence classes of ECAs with up to four rules in each class, each represented by its member with the lowest Wolfram number. The number of resulting equivalence classes is 88, the ECAs in each class having equivalent dynamics, as described in detail in [17].

We note that if two ECAs are equivalent, it does not necessarily follow that they can simulate one another; the problem is the left-right input bit symmetry, which requires a global transformation of the space-time diagrams to show the equivalence of two ECAs' dynamics. We could easily generalize our definition of enc–dec simulation to allow for such geometrical transformations (i.e., reflection of space-time diagrams) to account for this fact. However, for conciseness, we chose to keep the definition simple.

For better orientation, we show the results only for the 88 unique ECAs with respect to their equivalence classes. We write $\mathrm{ca}_1 \lesssim \mathrm{ca}_2$ if there is a ca_1' equivalent to ca_1 and ca_2' equivalent ca_2 such that $\mathrm{ca}_1' \leq \mathrm{ca}_2'$. The results can be plotted as a graph, each relation $\mathrm{ca}_1 \lesssim \mathrm{ca}_2$ represented as:



This representation gives us a hierarchy, with the "computationally weakest" ECAs at the bottom, and the "strongest" ones at the top.

5.2 Overview of Results

Using the search methods described previously, we have searched for relations with encoder sizes ranging from 2 to 24. Between the 88 unique ECAs, we found 548 relations in total, out of $88 \cdot 88 = 7744$ possible ones. It is,

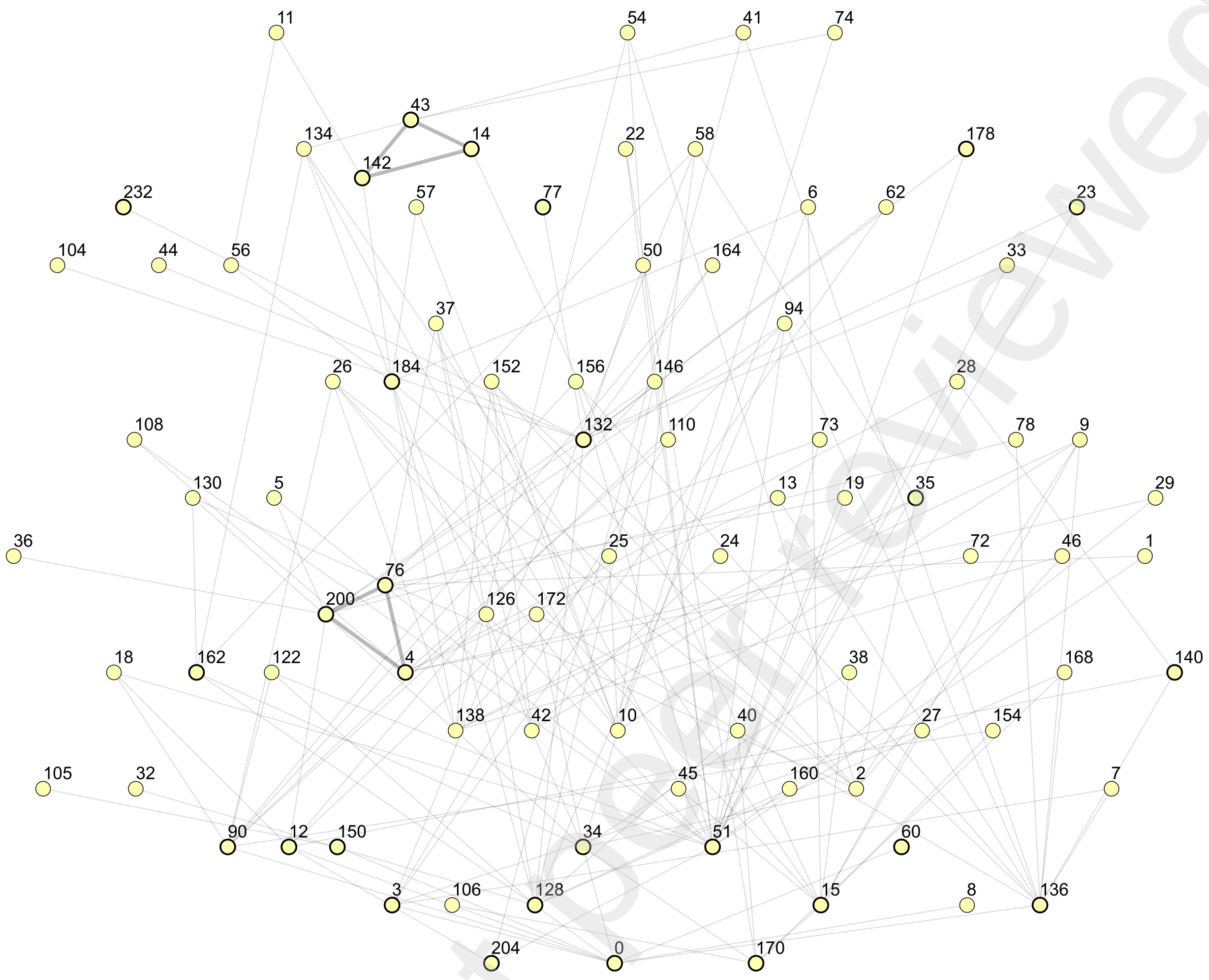


Figure 12: Resulting hierarchy of simulation relations we found between the 88 unique ECAs. The thicker edges connect computationally equivalent ECAs.

however, clear that this upper bound cannot be reached as e.g., ECA 0 cannot simulate anything other than itself.

We give the complete list of results in the Appendix, Table 3 and we illustrate the hierarchy in Figure 12. As a complementary material, we have implemented a JavaScript with an interactive version of the graph[2], which helps to better visualize its structure. The reader can explore the particular relations and their visualization in a Google Colab notebook[3] we created.

From Figure 13, we see that with increasing encoder size, the number of new relations we could find decreases. We however note that even for encoder of size 24 we were able to find some new relations. As we believe that our search methods can be further optimized, we encourage interested readers to explore the space of larger encoders.

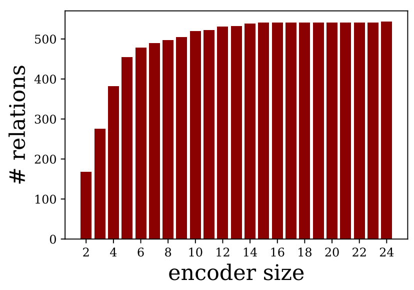


Figure 13: Cumulative plot of the total number of ECA relations found as the size of the encoder is increased.

In the rest of this section, we highlight certain parts of the resulting hierarchy that we found of importance.

[2]https://observablehq.com/d/62e715c58cd006a0

[3]https://bit.ly/eca_sim_google_colab

5.3 Computational Building Blocks

The resulting relations have a particular structure: there are a few ECAs that many others can simulate. Such ECAs, all at the bottom of the resulting hierarchy, are listed in Table 1.

ECA enc–dec simulation results, bottom of the hierarchy			
ECA	**#ECAs simulating**	**ECA**	**#ECAs simulating**
0	81 (92 %)	128	36 (41 %)
204	41 (47 %)	4	32 (36 %)
170	39 (44 %)	76	32 (36 %)
12	37 (42 %)	200	32 (36 %)

Table 1: ECAs from the bottom of the hierarchy that can be simulated by many other ECAs.

In this section, we show that such ECAs can be interpreted as basic computational building blocks and that this view helps us uncover the computational structure of cellular automata. We give some examples below.

Information Storage The global rule of 204 is the identity mapping, see Figure 14. Thus, any ECA that can simulate it is able to preserve information (under a suitable encoding) perfectly.

Information Passing The global rule of ECA 170 shifts the information "one bit to the left", see Figure 14. Therefore, any ECA able to simulate it can preserve information while shifting it in space.

Time Counter ECA 128 can be interpreted as a local "time counter". Its local rule is a ternary AND function. Thus, any consecutive block of 1's decays over time. If the block consists of $2k + 1$ 1's, then the block diminishes at time step $k + 1$, see Figure 14.

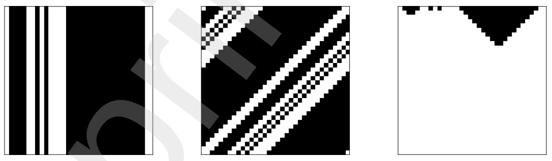


Figure 14: Space-time diagrams of ECA 204 (left), ECA 170 (middle), and ECA 128 (right).

Edge Detection We found that ECAs 4, 76, and 200 can simulate one another and are, therefore, computationally equivalent. By observing the simulation between them, we noticed that the ECAs' computational ability could be characterized as "edge detection". We illustrate this informal notion in Figure 15. When observing

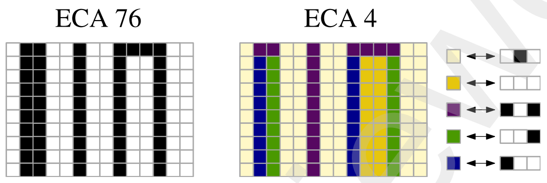


Figure 15: Compressed scheme of ECA 4 simulating ECA 76 with an encoder of size 3. In the right diagram, triples of cells are represented by different colors. We can see that on this richer state space, ECA 4 recognizes for each consecutive block of purple states whether each cell is the left edge of the block, the right edge, both, or whether it is in the middle.

the remaining pair-wise relations between ECAs 4, 76, and 200 we saw that analogous edge detecting diagrams are formed. It is interesting to note that this computational capacity is not easily noticeable when observing the space-time diagrams of the individual rules. We argue that studying the larger scale dynamics of CAs can thus help us uncover new structure in their dynamics.

Majority Computation In [3], it has been shown that ECA 184 can solve the task of computing the majority of 0's and 1's in its initial configuration. ECA 184 is not in the list of most frequently simulated automata, yet there are still 10 ECAs that can simulate it, and therefore, under a suitable encoding, such ECAs can solve the majority task as well.

As an example, ECA 41, which is at the top of the hierarchy, can implement four tasks described above: information storage and passing, time counting, and majority computation. An interesting open problem is how to compose such computational blocks within the dynamics of an automaton to perform harder tasks.

Interpreting the computations of more complex CAs is much more challenging. Nevertheless, we hypothesize that with identifying enough "computational building blocks" and with deep enough insight into the simulation relations between ECAs, the computational power of complex automata could potentially be fully characterized by the list of simpler computational building blocks they can simulate.

5.4 Reducing the Number of Unique ECAs

Another specificity of the structure of resulting relations is that there are many ECAs that cannot be simulated by any other ECA. There are 52 such automata, and we list them here: 1, 5, 6, 7, 8, 9, 11, 13, 18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 32, 33, 36, 37, 38, 40, 41, 44, 45, 46, 54, 57, 58, 62, 72, 73, 74, 78, 94, 104, 105, 106, 108, 110, 122, 126, 130, 152, 154, 156, 160, 164, 168, 172.

Suppose we have two ECAs such that $\mathrm{ca}_1 \lesssim \mathrm{ca}_2$. Then, we know the dynamics of ca_1 is contained in the larger scale dynamics of ca_2. Therefore, if we are interested in the variety of dynamics given by the class of ECAs, the dynamics of ca_1 is redundant. In this way, we argue that we can further reduce the 88 equivalence classes of ECAs to just 52 classes that have unique dynamics.

This type of reduction is useful, for example, if we wish to search for automata capable of solving a given complex task. We have a guarantee that the 33 omitted ECAs cannot compute anything more than the 52 new representatives. In another words, by this reduction, we have not lost any computational capacity of ECAs. This type of reduction could be particularly useful for larger CA spaces.

5.5 Computationally Equivalent ECAs

We have found two triples of ECAs that are computationally equivalent. One triple is the “edge detecting" automata 4, 76, and 200 at the bottom of the hierarchy.

The second equivalent triple is formed by ECAs 14, 43, and 142. We claim this equivalence is of more interest as the dynamics of these ECAs are more complex. Indeed, from our results, the automata are amongst the most computationally powerful ones as they can each simulate 12 ECAs (see Table 2). There have been efforts to study the global dynamics of ECA 14 ([24], [12]), and the benefit of the newly discovered computational equivalence is that some of the results could be extended to the equivalent ECAs in a straightforward way.

We argue that the main significance of the newly discovered computational equivalence is that it reduces the number of ECAs with unique computational capacity. We hypothesize that by making the search methods of the relations more effective and the notion of simulation more general, the number of computationally equivalent ECAs could be further increased. Such results would support the bold claim of Stephen Wolfram stated in [29] as the Principle of Computational Equivalence: “Almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication". Viewing this in the context of cellular automata, the relative simulation of CA is a framework that helps us rigorously show the computational equivalence of such systems.

5.6 Most Computationally Powerful ECAs

From our results, the ECAs that can simulate the most automata are represented in Table 2.

ECA enc–dec simulation results, top of the hierarchy	
ECA	**ECAs simulated (count)**
41	0, 2, 10, 15, 34, 42, 128, 134, 136, 138, 162, 170, 184, 204 **(14)**
54	0, 2, 4, 12, 34, 50, 51, 76, 128, 132, 138, 170, 200, 204 **(14)**
11	0, 2, 14, 15, 34, 42, 43, 56, 128, 138, 142, 170, 184 **(13)**
74	0, 2, 10, 14, 15, 34, 42, 43, 128, 138, 142, 170, 184 **(12)**
14, 43, 142	0, 2, 14, 15, 34, 42, 43, 142, 128, 138, 142, 170, 184 **(12)**

Table 2: Most computationally powerful ECAs according to the relations we found. In the 5th row, ECAs 14, 42 and 142 can simulate an identical set of automata, thus we list them together for conciseness.

If we disregard the cycles in the hierarchy, i.e., we identify each set of computationally equivalent ECAs with a single node, we can measure the longest paths in the graph. There are multiple ones, all of length five, starting with ECA 11, 41, 54, or 74 as the most powerful automaton. We give four examples:

$$11 \gtrsim 56 \gtrsim 184 \gtrsim 138 \gtrsim 34 \gtrsim 0$$
$$41 \gtrsim 134 \gtrsim 184 \gtrsim 42 \gtrsim 34 \gtrsim 0$$
$$54 \gtrsim 50 \gtrsim 132 \gtrsim 200 \gtrsim 12 \gtrsim 0$$
$$74 \gtrsim 14 \gtrsim 184 \gtrsim 42 \gtrsim 34 \gtrsim 170$$

5.7 Limitations of Additive ECAs

Let $\oplus$ denote the binary XOR operation. An ECA is called *additive*, if its local rule f is of the form $f(x_1, x_2, x_3) = \alpha \cdot x_1 \oplus \beta \cdot x_2 \oplus \gamma \cdot x_3$ for some $\alpha, \beta, \gamma \in \mathbf{2}$. This property makes such CAs amenable to algebraic studies, and indeed, many results about their dynamics have been proven ([18]). There are six such unique ECAs: 0, 60, 90, 150, 170, and 204. When observing the hierarchy of relations, we can notice that additive ECA are able to simulate only themselves and other additive ECAs. Indeed, in [2] the authors have shown that additive one-dimensional CAs with two states are self-similar.

Regarding previous work on CA simulation, it can be easily shown that additive CAs can only coarse-grain to additive CAs. Similarly, it is straightforward that additive CAs can only contain additive CAs as subautomata ([19]). We hypothesize that this limitation will also be preserved with the enc–dec simulation: that additive CAs are only capable to enc–dec simulate additive CAs. It is out of the scope of this paper to study such results. However, we note it as a goal for our future work.

5.8 Computation and Chaos

In [11] we have introduced the notion of computational chaos motivated by the following observation. Let ca_1 and ca_2 be CAs such that $ca_1 \leq_k \mathrm{ca}_2$ for some $k \in \mathbb{N}$. This simply means that by aggregating blocks of k cells into new, larger cells and letting the ca_2 operate on them by iterating its global rule k times, we get a new larger scale automaton that contains the dynamics of ca_1. This means that a part of large scale dynamics of ca_2 can be described by the much simpler iterations of ca_1. In other words, the large scale dynamics of ca_2 contain some structure that can be described simply by iterating ca_1. Intuitively, a CA has chaotic dynamics if no large scale of its behavior can be described in simpler terms. This motivated our definition of *computational chaos*. We say that a CA is computationally chaotic if it cannot simulate any CA non-trivially (i.e., with an encoding of size $k \geq 2$).

From our results, it follows that the only ECA that possibly cannot simulate any other automaton, nor itself non-trivially, is ECA 30. Therefore, it motivates the hypothesis that ECA 30 (and its equivalent counterparts) is the only computationally chaotic ECA. ECA 30 is classically considered as a CA with chaotic dynamics. It is of great interest whether our limited search of encoders up to size 24 failed to find a counterexample for ECA 30, or whether this automaton possesses some fundamental properties which makes every scale of its dynamics unamenable for concise description.

6 Conclusion

In this paper, we have introduced a novel method of comparing the computational capacity of cellular automata that we call the enc–dec simulation. Informally, ca_2 can enc–dec simulate ca_1 if the larger scale dynamics of ca_2 “contain" the dynamics of ca_1. This method generalizes previous approaches and helps us identify a rich structure of relationships between cellular automata. We demonstrated this on the class of elementary CAs. We found 548 relations in total between the 88 classes of unique ECAs. We also argued that the space of ECAs with unique dynamics can be reduced from 88 to 52. Moreover, we identified the two triples (ECAs 4, 76, 200 and ECAs 14, 43, and 142) that are computationally equivalent. We believe that using similar approaches, the number of unique automata can be further reduced in the future. Such reductions can be especially important in large automata spaces, such as 2D automata with multiple states.

6.1 Generalizations and Future Work

We introduced the enc–dec simulation for arbitrary one-dimensional cellular automata, but we note that this notion can be generalized to higher dimensions as well. As long as one determines the shape of the larger, aggregated cells (e.g., a k by k square in 2D automata), one can introduce the notion of larger scale dynamics and the simulation relation analogously.

There are multiple ways to generalize the enc–dec simulation further by imposing fewer restrictions on the enc and dec mappings. For example, one can allow the mappings to do some basic geometrical transformations of the configuration space, such as shifts or reflections ([8]). We could also allow the “stronger" CA to simulate only every m-th step of the “weaker" CA, $m \in \mathbb{N}$ ([20]). It would be of particular interest to see whether such generalizations would reduce the space of ECAs dramatically by identifying more computationally equivalent automata.

A possible future project is to devise more effective methods of searching for good candidate encoder–decoder pairs. We hypothesize that in large encoder spaces, it might be possible to find relations between automata with complex dynamics, which would be of particular interest. For example, it has been elaborately proven in [5] that ECA 110 is Turing complete. By finding ECAs that can simulate ECA 110, we would have a simple way of showing their Turing completeness.

Recently, there has been fascinating research developing around cellular automata with continuous state space ([21], [23], [4], [28]). It would be interesting to generalize the concept of CA simulation for such automata. The encoder and decoder mappings could be represented by simple neural networks, and searching for a relation between two continuous CAs could be done with more effective methods, such as the standard gradient descent.

7 Acknowledgments

Our work was supported by Grant Schemes at CU, reg. no. CZ.02.2.69/0.0/0.0/19_073/0016935, by SVV-2020-260589, by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund, and is part of the RICAIP project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 857306. This work was partially financed by the Research Council of Norway's DeepCA project, grant agreement 286558.

We would like to thank Jiří Tůma, Jakub Krásenský, Tom Glover, and Trym Lindell for inspiring discussions and valuable feedback.

8 Appendix

In the Appendix, we prove that the enc–dec simulation forms a preorder, we present a criterion for verifying that an enc–dec pair witnesses a particular relation, and we give the complete list of relations we found for ECAs.

8.1 CA Enc-Dec Simulation Forms a Preorder

The simulation relation $\leq$ is clearly reflexive, as for each ca, it holds that $\mathrm{ca} \leq_1 \mathrm{ca}$ with the witnessing encoding and decoding being identity mappings. In this section, we show that $\leq$ is also transitive and, therefore, that it forms a preorder.

CA simulation relation is transitive

Observation 1. Let $\mathrm{ca}_1 = (S, f)$, $\mathrm{ca}_2 = (T, g)$, and $\mathrm{ca}_3 = (U, h)$ be 1D cellular automata with global rules F, G, and H respectively. Let us suppose that:

$$\mathrm{ca}_1 \leq_k ca_2 \text{ with } \mathrm{enc}_1 : S \to T^k \text{ and } \mathrm{dec}_1 : T^k \to S$$
$$\mathrm{ca}_2 \leq_l \mathrm{ca}_3 \text{ with } \mathrm{enc}_2 : T \to U^l \text{ and } \mathrm{dec}_2 : U^l \to T.$$

Then, $\mathrm{ca}_1 \leq_{kl} ca_3$.

Proof. Let $s \in S$. We define $\mathrm{enc} : S \to U^{kl}$ by:

$$\mathrm{enc}(s) = \mathrm{enc}_2\big(\mathrm{enc}_1(s)_0\big) \dots \mathrm{enc}_2\big(\mathrm{enc}_1(s)_{k-1}\big).$$

Similarly, let $u_0, u_1, \dots, u_{k-1} \in U^l$. We define the decoder $\mathrm{dec} : U^{kl} \to S$ by:

$$\mathrm{dec}(u_0 \dots u_{k-1}) = \mathrm{dec}_1\big(\mathrm{dec}_2(u_0) \dots \mathrm{dec}_2(u_{k-1})\big).$$

We wil show that enc and dec witness $\mathrm{ca}_1 \leq_{kl} ca_3$.

Let $\varphi_1 : (T^k)^{\mathbb{Z}} \to T^{\mathbb{Z}}$ be the natural bijection given by $\varphi(c)_{ki}\varphi(c)_{ki+1} \dots \varphi(c)_{ki+k-1} = c_i \in T^k$ for each $c \in (T^k)^{\mathbb{Z}}$ and $i \in \mathbb{Z}$. Analogously, we define the bijection $\varphi_2 : (U^{kl})^{\mathbb{Z}} \to (U^l)^{\mathbb{Z}}$. Then, it is straightforward to verify that

$$\overline{\mathrm{enc}} = \varphi_2^{-1} \circ \overline{\mathrm{enc}_2} \circ \varphi_1 \circ \overline{\mathrm{enc}_1},$$
$$\overline{\mathrm{dec}} = \overline{dec_1} \circ \varphi_1^{-1} \circ \overline{\mathrm{dec}_2} \circ \varphi_2.$$

Let $c \in S^{\mathbb{Z}}$ and $t \in \{0, 1, \dots\}$. We have:

$$\begin{aligned} F^t &= \overline{\mathrm{dec}_1} \circ G_k^t \circ \overline{\mathrm{enc}_1} \\ &= \overline{\mathrm{dec}_1} \circ \varphi_1^{-1} \circ G^{kt} \circ \varphi_1 \circ \overline{\mathrm{enc}_1} \\ &= \overline{\mathrm{dec}_1} \circ \varphi_1^{-1} \circ \overline{\mathrm{dec}_2} \circ H_l^{kt} \circ \overline{\mathrm{enc}_2} \circ \varphi_1 \circ \overline{\mathrm{enc}_1} \\ &= \overline{\mathrm{dec}_1} \circ \varphi_1^{-1} \circ \overline{\mathrm{dec}_2} \circ \varphi_2 \circ H_{kl}^t \circ \\ &\quad \circ \varphi_2^{-1} \circ \overline{\mathrm{enc}_2} \circ \varphi_1 \circ \overline{\mathrm{enc}_1} \\ &= \overline{\mathrm{dec}} \circ H_{kl}^t \circ \overline{\mathrm{enc}}. \end{aligned}$$

□

8.2 CA Large scale dynamics: Algebraic motivation

When defining CA enc–dec simulation, we introduced the notion of large scale CA dynamics; it was defined as follows. Let $\mathrm{ca} = (T, g)$ be a 1D CA with global rule F and let $k \in \mathbb{N}$. Let $\varphi : (T^k)^{\mathbb{Z}} \to T^{\mathbb{Z}}$ be the natural bijection. We defined the larger scale mapping $G_k : (T^k)^{\mathbb{Z}} \to (T^k)^{\mathbb{Z}}$ as $G_k = \varphi^{-1} \circ G^k \circ \varphi$. In this section, we show that in fact, there is an automaton ca^k whose global function is exactly G_k. Thus, ca^k is an automaton describing the larger scale dynamics of ca. Let us first introduce some terminology.

Terminology Let T be a finite non-empty set. We define $T^+ = \bigcup_{k=1}^{\infty} T^k$ to be the set of all finite nonempty sequences, and $T^* = T^+ \cup \{\emptyset\}$. A bi-infinite sequence v over T is a mapping from $\mathbb{Z}$ to T, and we write $v = \dots v_{-2}v_{-1}v_0v_1v_2\dots$, $v_i \in T$ for all $i \in \mathbb{Z}$. Let $u = u_0 \dots u_{n-1} \in T^n$ and $v = v_0 \dots v_{m-1} \in T^m$. We define the concatenation of u and v to be the sequence $u_0u_1 \dots u_{n-1}v_0v_1 \dots v_{m-1}$, which we denote by uv. We say that v contains u if $v = lur$ for some $l, r \in T^*$. Analogously, for $u \in T^n$, $n \in \mathbb{N}$, and $v \in T^{\mathbb{Z}}$ we say that v contains u if there exists some $i \in \mathbb{Z}$ such that $v_iv_{i+1} \dots v_{i+n-1} = u$.

For simplicity of notation, we will focus solely on ECAs in this section. However, the results can be generalized to arbitrary 1D CAs in a straightforward way.

Definition 2 (Unravelling a Boolean function)**.**
Let $g : \mathbf{2}^3 \to \mathbf{2}$. We define $\widetilde{g} : \mathbf{2}^+ \to \mathbf{2}^+$ as

$$\widetilde{g}(b_0, \dots, b_{n-1}) = g(b_0, b_1, b_2) \dots g(b_{n-3}, b_{n-2}, b_{n-1})$$

for any binary sequence $b_0b_1 \dots b_{n-1}$, $n \geq 3$.

By $\widetilde{g}^k$ we simply mean the composition

$$\widetilde{g}^k = \underbrace{\widetilde{g} \circ \widetilde{g} \circ \dots \circ \widetilde{g}}_{k-\text{times}}.$$

Each iteration of $\widetilde{g}$ shortens the input size by 2, therefore we can notice that when we restrict the domain of $\widetilde{g}^k$ to the set $\mathbf{2}^{3k}$ we get

$$\widetilde{g}^k|_{\mathbf{2}^{3k}} : \mathbf{2}^k \times \mathbf{2}^k \times \mathbf{2}^k \to \mathbf{2}^k.$$

Thus, $g_k := \widetilde{g}^k|_{\mathbf{2}^{3k}}$ can be interpreted as a ternary function operating on binary sequences of size k. We show how g_k is computed in Example 3.

Example 3. Let g be the local rule of ECA 56. Its table is represented below in Figure 16.

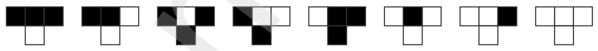


Figure 16: ECA 56 rule table.

In Figure 17 we show a simple example of how to compute $g_3(010, 001, 110)$.

Thus, we have a new automaton $\text{ca}^k = (\mathbf{2}^k, g_k)$ which represents the large scale dynamics of the CA $\text{ca} = (\mathbf{2}, g)$. It is straightforward to verify that the global function of ca^k is exactly G_k.

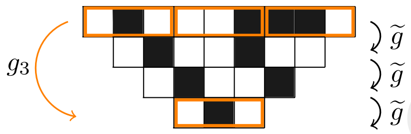


Figure 17: Scheme of g_3 computing on blocks of size 3.

Let $\text{ca}_1 = (\mathbf{2}, f)$, $\text{ca}_2 = (\mathbf{2}, g)$ be two ECAs for which $\text{ca}_1 \leq_k \text{ca}_2$. Then, this informally means that the dynamics of ca_1 is contained in the dynamics of ca_2^k. Thus, studying whether $\text{ca}_1 \leq_k \text{ca}_2$ reduces to studying the relationship between the local rules f and g_k. We use this "localization" to introduce the Verification algorithm below.

8.3 Verification Algorithm

For the whole section, let us fix ECAs $\text{ca}_1 = (\mathbf{2}, f)$, $\text{ca}_2 = (\mathbf{2}, g)$ with global rules F and G respectively.

In this section, we present a criterion that can be efficiently verified on a computer, and that has the following property. Whenever an $\text{enc} : \mathbf{2} \to \mathbf{2}^k$ and $\text{dec} : \mathbf{2}^k \to \mathbf{2}$ satisfy the criterion, we have a proof that $\text{ca}_1 \leq_k \text{ca}_2$.

Definition 4. (Forbidden blocks) Let $\text{enc} : \mathbf{2} \to \mathbf{2}^k$ and $\text{dec} : \mathbf{2}^k \to \mathbf{2}$. Let $c_1, c_2, c_3 \in \mathbf{2}^k$ we say that the sequence $c_1c_2c_3$ is a *forbidden block for* ca_1, ca_2, enc, *and* dec if: $b_1b_2b_3 = \text{dec}(c_1)\text{dec}(c_2)\text{dec}(c_3)$, $b = f(b_1, b_2, b_3)$, $c = g_k(c_1, c_2, c_3)$ and $b \neq \text{dec}(c)$. This is illustrated in Figure 18.

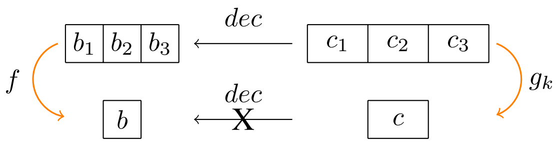


Figure 18: Scheme of a forbidden block.

Suppose we want to verify that an enc–dec pair witnesses the relation $\text{ca}_1 \leq_k \text{ca}_2$. This is equivalent to verifying that in the space-time diagrams of ca_2^k started from encoded configurations, we never "encounter" a forbidden block. We formalize this below.

Definition 5. (Image of CA under encoder) Let S and T be finite nonempty sets, let $e : S \to T$ be a mapping with image $e(S) = T' \subseteq T$, and let (T, g) be a CA with global rule G. We define the *image of CA* (T, g) *of configurations encoded by* e as

$$\text{Im}_e(G) = \{G^t(c) \,|\, c \in (T')^{\mathbb{Z}},\, t \in \{0, 1, 2, \dots\}\}.$$

Let us consider enc : $\mathbf{2} \to \mathbf{2}^k$ and dec : $\mathbf{2}^k \to \mathbf{2}$ and suppose that we want to prove that the enc, dec pair witnesses the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. Then, the encoder determines the image $\mathrm{Im}_{\mathrm{enc}}(G_k) \subseteq (\mathbf{2}^k)^{\mathbb{Z}}$. $\mathrm{Im}_{\mathrm{enc}}(G_k)$ is exactly the subset of configurations for which we have to check they are decoded correctly to correspond to the dynamics of ca_1. This is formalized in Observation 2.

Observation 2. Let enc : $\mathbf{2} \to \mathbf{2}^k$ and dec : $\mathbf{2}^k \to \mathbf{2}$. Then, enc, dec witness the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ if and only if no configuration in $Im_{\mathrm{enc}}(G_k)$ contains any forbidden block for ca_1, ca_2, enc, and dec.

If we could efficiently compute $Im_{\mathrm{enc}}(G_k)$, such verification would be straightforward. This is, however, not always the case as the set $Im_{\mathrm{enc}}(G_k)$ might have no compact description. Instead, we will approximate it with a set M, $M \supseteq Im_{\mathrm{enc}}(G_k)$, which can be computed efficiently. Then, we check that no forbidden block is contained in any configuration of M. If this is the case, then we have the proof that the given *enc*, *dec* pair indeed witnesses the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. Below, we describe the construction of M.

Approximating $Im_{\mathrm{enc}}(G_k)$

The core of the idea is that we "locally approximate $Im_{\mathrm{enc}}(G_k)$ by triples of cells". Let $\mathrm{ca} = (T, g)$ be a CA with radius 1 and global rule G, and let $M_3 \subseteq T^3$. We say that M_3 is *closed under G* if the following condition holds: for every $b \in T^5$, $b = b_0b_1b_3b_3b_4$ such that $b_0b_1b_2$, $b_1b_2b_3$, $b_2b_3b_4 \in M_3$ it holds that $\widetilde{g}(b) \in M_3$. This notion is illustrated in Figure 19.

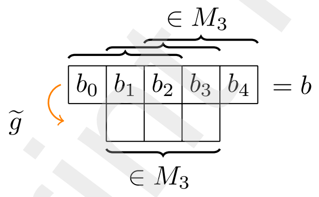


Figure 19: Scheme of a forbidden block.

Let $M_3 \subseteq T^3$ be closed under G and let $\mathcal{M}_3 = \{c \in T^{\mathbb{Z}} \,|\, c_{i-1}c_ic_{i+1} \in M_3 \text{ for all } i \in \mathbb{Z}\}$. Then, we show that $G(\mathcal{M}_3) \subseteq \mathcal{M}_3$.

Proof. Let $c \in \mathcal{M}_3$. We show that $G(c) \in \mathcal{M}_3$. Let $i \in \mathbb{Z}$. It suffices to show that $G(c)_{i-1}G(c)_iG(c)_{i+1} \in M_3$; $G(c)_{i-1}G(c)_iG(c)_{i+1} = \widetilde{g}(c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2})$. Since $c_{i-2}c_{i-1}c_i$, $c_{i-1}c_ic_{i+1}$, $c_ic_{i+1}c_{i+2} \in M_3$ and M_3 is closed under G, we have that $\widetilde{g}(c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}) \in M_3$. □

In the particular case of enc–dec simulations of ECAs, the encoder defines a subset $\{\mathrm{enc}(0), \mathrm{enc}(1)\} \subseteq 2^k$ of the state space of the CA $(\mathbf{2}^k, g_k)$ with global rule G_k. Our goal is to approximate $Im_{\mathrm{enc}}(G_k)$ by triples of cells. We would like to construct a set $M_3 \subseteq (\mathbf{2}^k)^3$ such that $\{\mathrm{enc}(0), \mathrm{enc}(1)\}^3 \subseteq M_3$ and such that M_3 is closed under G_k. Then, it follows that $\mathcal{M}_3 = \{c \in T^{\mathbb{Z}} \,|\, c_{i-1}c_ic_{i+1} \in M_3 \,\forall i \in \mathbb{Z}\} \supseteq Im_{\mathrm{enc}}(G_k)$. The construction is quite straightforward. We "start" with the set $\{enc(0), enc(1)\}^3$ and we "enclose" it under G_k by adding all the images necessary. The details are described in Algorithm 3.

Algorithm 3: Algorithm Approximating $Im_{\mathrm{enc}}(G_k)$.

```
Input  : function $g_k$, $\mathrm{enc}(0), \mathrm{enc}(1) \in \mathbf{2}^k$
Output: set $M_3$ closed under $G_k$ containing
        $\{enc(0), enc(1)\}^3$
1  $M := \{enc(0), enc(1)\}^3$
2  m_closed=False
3  while not m_closed do
4      m_closed=True
5      for all $b = b_0b_1b_2b_3b_4 \in (\mathbf{2}^k)^5$ do
6          if $b_0b_1b_2$, $b_1b_2b_3$, $b_2b_3b_4 \in M$ then
7              if $\widetilde{g_k}(b) \notin M$ then
8                  add $\widetilde{g_k}(b)$ to $M$
9                  m_closed=False
10             end
11         end
12     end
13 end
```

Finally, we can describe the criterion for verifying the correctness of an enc–dec simulation of ECAs.

Algorithm 4: Verification Algorithm for $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$.

```
Input  : $ca_1 = (\mathbf{2}, f)$, $ca_2 = (\mathbf{2}, g)$, $enc : \mathbf{2} \to \mathbf{2}^k$, and
         dec : $\mathbf{2}^k \to \mathbf{2}$
Output: proof that $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ or an "unsuccessful"
         message.
1  1. compute the set of forbidden blocks $F \subseteq (2^k)^3$ for $\mathrm{ca}_1$,
      $\mathrm{ca}_2$, enc, dec.
2  2. using Algorithm 3 compute the set $M_3 \subseteq (2^k)^3$ which
      approximates $Im_{\mathrm{enc}}(G_k)$
3  3. If $F \cap M_3 = \emptyset$ then return $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$ verified
4     Else return "unsuccessful"
```

If Algorithm 4 returns an unsuccessful message, we cannot conclude that the given enc, dec do not witness the relation $\mathrm{ca}_1 \leq_k \mathrm{ca}_2$. It might be that our approximation of $Im_{\mathrm{enc}}(G_k)$ by triples of cells is too coarse. We can refine it by approximating it instead by m-tuples of cells with $m \geq 4$ in an analogous way. It is out of the scope of this paper to describe such methods in detail.

8.4 Complete Results

In this section, we present all the simulation relations we found between the 88 unique ECAs when searching through encoders of sizes 2 to 24. All the relations have been verified by the Verification Algorithm (4).

ECA enc–dec simulation complete results	
ECA	**ECAs simulated (count)**
0	0 (**1**)
1	0, 4, 12, 51, 76, 200, 204 (**7**)
2	0, 34, 170 (**3**)
3	0, 3 (**2**)
4	0, 4, 12, 76, 200, 204 (**6**)
5	0, 4, 12, 51, 76, 200, 204 (**7**)
6	0, 2, 10, 15, 34, 42, 128, 138, 170, 184 (**10**)
7	0, 3, 136 (**3**)
8	0 (**1**)
9	0, 2, 10, 15, 34, 136, 138, 170 (**8**)
10	0, 34, 170 (**3**)
11	0, 2, 14, 15, 34, 42, 43, 56, 128, 138, 142, 170, 184 (**13**)
12	0, 12, 204 (**3**)
13	0, 4, 12, 76, 136, 200, 204 (**7**)
14	0, 2, 14, 15, 34, 42, 43, 128, 138, 142, 170, 184 (**12**)
15	15, 170 (**2**)
18	0, 12, 51, 90, 204 (**5**)
19	0, 4, 12, 51, 76, 200, 204 (**7**)
22	0, 4, 12, 51, 76, 90, 128, 146, 170, 200, 204 (**11**)
23	0, 4, 12, 23, 51, 76, 128, 132, 200, 204 (**10**)
24	0, 34, 42, 170 (**4**)
25	0, 10, 34, 138, 170 (**5**)
26	0, 2, 10, 15, 34, 90, 138, 170 (**8**)
27	0, 3, 15, 170 (**4**)
28	0, 4, 12, 51, 76, 136, 140, 200, 204 (**9**)
29	0, 4, 12, 51, 76, 200, 204 (**7**)
30	- (**0**)
32	0, 128 (**2**)
33	0, 4, 12, 51, 76, 128, 132, 200, 204 (**9**)
34	0, 34, 170 (**3**)
35	0, 2, 34, 35, 42, 138, 170 (**7**)
36	0, 4, 12, 76, 200, 204 (**6**)
37	0, 4, 12, 34, 42, 51, 76, 128, 170, 200, 204 (**11**)
38	0, 15, 34, 170 (**4**)
40	0, 128, 170 (**3**)
41	0, 2, 10, 15, 34, 42, 128, 134, 136, 138, 162, 170, 184, 204 (**14**)
42	0, 34, 170 (**3**)
43	0, 2, 14, 15, 34, 42, 43, 128, 138, 142, 170, 184 (**12**)
44	0, 4, 12, 76, 128, 132, 200, 204 (**8**)
45	15, 170 (**2**)
46	0, 2, 34, 138, 170 (**5**)
50	0, 4, 12, 51, 76, 128, 132, 200, 204 (**9**)
51	51, 204 (**2**)
54	0, 2, 4, 12, 34, 50, 51, 76, 128, 132, 138, 170, 200, 204 (**14**)
56	0, 2, 34, 42, 128, 138, 170, 184 (**8**)
57	0, 2, 10, 34, 42, 128, 138, 170, 184 (**9**)
58	0, 2, 3, 10, 34, 35, 42, 128, 138, 162, 170 (**11**)
60	0, 60 (**2**)
62	0, 3, 4, 12, 76, 128, 132, 200, 204 (**9**)
72	0, 4, 12, 76, 200, 204 (**6**)
73	0, 4, 12, 51, 76, 200, 204 (**7**)
74	0, 2, 10, 14, 15, 34, 42, 43, 128, 138, 142, 170, 184 (**13**)
76	0, 4, 12, 76, 200, 204 (**6**)
77	0, 4, 12, 76, 77, 128, 132, 200, 204 (**9**)
78	0, 4, 12, 76, 136, 200, 204 (**7**)
90	0, 90 (**2**)
94	0, 4, 12, 51, 76, 90, 128, 136, 200, 204 (**10**)
104	0, 4, 12, 76, 128, 132, 200, 204 (**8**)
105	0, 150 (**2**)
106	170 (**1**)
108	0, 4, 12, 51, 76, 200, 204 (**7**)
110	0, 3, 4, 12, 76, 200, 204 (**7**)
122	0, 51, 90, 128, 204 (**5**)
126	0, 12, 51, 90, 204 (**5**)
128	0, 128 (**2**)
130	0, 2, 34, 128, 138, 162, 170 (**7**)
132	0, 4, 12, 76, 128, 132, 200, 204 (**8**)
134	0, 2, 10, 15, 34, 42, 128, 138, 162, 170, 184 (**11**)
136	0, 136 (**2**)
138	0, 34, 170 (**3**)
140	0, 12, 136, 140, 204 (**5**)
142	0, 2, 14, 15, 34, 42, 43, 128, 138, 142, 170, 184 (**12**)
146	0, 4, 12, 51, 76, 90, 128, 200, 204 (**9**)
150	0, 150 (**2**)
152	0, 2, 10, 34, 42, 136, 138, 170 (**8**)
154	0, 15, 90, 170 (**4**)
156	0, 4, 12, 51, 76, 136, 200, 204 (**8**)
160	0, 128 (**2**)
162	0, 34, 128, 162, 170 (**5**)
164	0, 4, 12, 76, 90, 128, 132, 200, 204 (**9**)
168	0, 128, 136, 170 (**4**)
170	170 (**1**)
172	0, 12, 34, 136, 170, 204 (**6**)
178	0, 4, 12, 51, 76, 128, 132, 178, 200, 204 (**10**)
184	0, 2, 34, 42, 128, 138, 170, 184 (**8**)
200	0, 4, 12, 76, 200, 204 (**6**)
204	204 (**1**)
232	0, 4, 12, 76, 128, 132, 200, 204, 232 (**9**)

Continued in the subsequent column

Table 3: List of all the 88 unique ECAs together with all the ECAs each can simulate and the total count in the parentheses.

References

[1] S. Bandini, G. Mauri, and R. Serra. “Cellular automata: From a theoretical parallel computational model to its application to complex systems”. In: *Parallel Computing* 27.5 (2001). Cellular automata: From modeling to applications, pp. 539–553. ISSN: 0167-8191. DOI: https://doi.org/10.1016/S0167-8191(00)00076-4. URL: https://www.sciencedirect.com/science/article/pii/S0167819100000764.

[2] A. Barbé et al. “Coarse-Graining Invariant Pattern of One-Dimensional Two-State Linear Cellular Automata”. In: Int. J. Bifurcation and Chaos (May 1995). DOI: 10.1142/S0218127495001216.

[3] Mathieu S. Capcarrere, Moshe Sipper, and Marco Tomassini. “Two-state, $r = 1$ Cellular Automaton that Classifies Density”. In: *Phys. Rev. Lett.* 77 (24 Dec. 1996), pp. 4969–4971. DOI: 10.1103/PhysRevLett.77.4969. URL: https://link.aps.org/doi/10.1103/PhysRevLett.77.4969.

[4] Bert Wang-Chak Chan. “Lenia and Expanded Universe”. In: *The 2020 Conference on Artificial Life*. MIT Press, 2020. DOI: 10.1162/isal_a_00297. URL: https://doi.org/10.1162%2Fisal_a_00297.

[5] Matthew Cook. “Universality in Elementary Cellular Automata”. In: *Complex Systems* 15 (Jan. 2004).

[6] James Crutchfield and Karl Young. “Inferring statistical complexity”. In: *Physical review letters* 63 (Aug. 1989), pp. 105–108. DOI: 10.1103/PhysRevLett.63.105.

[7] Karel Culik II and Shuangni Yu. “Undecidability of CA Classification Schemes. Complex Systems 2, 177-190”. In: *Complex Systems* 2 (Jan. 1988).

[8] Marianne Delorme et al. “Bulking I: an Abstract Theory of Bulking”. In: *Theoretical Computer Science* 412.30 (2011), pp. 3866–3880. DOI: 10.1016/j.tcs.2011.02.023. URL: https://hal.archives-ouvertes.fr/hal-00980376.

[9] Martin Gardener. “The fantastic combinations of John Conway’s new solitaire game “life” by Martin Gardner”. In: *Scientific American* 223 (1970), pp. 120–123.

[10] James E. Hanson. “Emergent Phenomena in Cellular Automata”. In: *Meyers R. (eds) Encyclopedia of Complexity and Systems Science* (2009). DOI: 10.1007/978-0-387-30440-3_51. URL: https://doi.org/10.1007/978-0-387-30440-3_51.

[11] Barbora Hudcová and Tomáš Mikolov. “Computational Hierarchy of Elementary Cellular Automata”. In: ALIFE 2021: The 2021 Conference on Artificial Life (July 2021). 105. DOI: 10.1162/isal_a_00447. URL: https://doi.org/10.1162/isal_a_00447.

[12] Shuichi Inokuchi. “On Behaviours of Cellular Automata with Rule 14 and 142”. In: *Kyushu Journal of Mathematics* 54.1 (2000), pp. 111–125. DOI: 10.2206/kyushujm.54.111.

[13] Navot Israeli and Nigel Goldenfeld. “Coarse-graining of cellular automata, emergence, and the predictability of complex systems”. In: *Physical Review E* 73.2 (Feb. 2006). ISSN: 1550-2376. DOI: 10.1103/physreve.73.026203. URL: http://dx.doi.org/10.1103/PhysRevE.73.026203.

[14] Jarkko J. Kari. “Basic Concepts of Cellular Automata”. In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–24. DOI: 10.1007/978-3-540-92910-9_1. URL: https://doi.org/10.1007/978-3-540-92910-9_1.

[15] Christopher G. Langton. “Self-reproduction in cellular automata”. In: *Physica D: Nonlinear Phenomena* 10.1 (1984), pp. 135–144. ISSN: 0167-2789.

[16] Christopher G. Langton. “Studying artificial life with cellular automata”. In: *Physica D: Nonlinear Phenomena* (1986).

[17] Wentian Li and Norman H. Packard. “The Structure of the Elementary Cellular Automata Rule Space”. In: *Complex Syst.* 4 (1990).

[18] Olivier Martin, Andrew Odlyzko, and Stephen Wolfram. “Algebraic Properties of Cellular Automata”. In: *Communications in Mathematical Physics* 93 (June 1984). DOI: 10.1007/BF01223745.

[19] Jacques Mazoyer and Ivan Rapaport. "Additive cellular automata over $\mathbb{Z}_p$ and the bottom of (CA,$\leq$)". In: vol. 1450. Aug. 2006, pp. 834–843. ISBN: 978-3-540-64827-7. DOI: 10.1007/BFb0055835.

[20] Jacques Mazoyer and Ivan Rapaport. "Inducing an order on cellular automata by a grouping operation". In: *Discrete Applied Mathematics* 91 (Jan. 1999), pp. 177–196. DOI: 10.1016/S0166-218X(98)00125-5.

[21] Alexander Mordvintsev et al. "Growing Neural Cellular Automata". In: *Distill* (2020). https://distill.pub/2020/growing-ca. DOI: 10.23915/distill.00023.

[22] J. V. Neumann and A. W. Burks. "Theory of Self-Reproducing Automata". In: *University of Illinois Press* 9 (1966).

[23] Stefano Nichele et al. "CA-NEAT: Evolved Compositional Pattern Producing Networks for Cellular Automata Morphogenesis and Replication". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2018), pp. 687–700. DOI: 10.1109/TCDS.2017.2737082.

[24] Xiaofeng Liao Qi Han and Chuandong Li. "Complex Dynamic Behaviors in Cellular Automata Rule 14". In: *Discrete Dynamics in Nature and Society* 2012 (2012), p. 12. DOI: 10.1155/2012/258309.

[25] Sebastian Risi. "The Future of Artificial Intelligence is Self-Organizing and Self-Assembling". In: *sebastianrisi.com* (2021). URL: https://sebastianrisi.com/self_assembling_ai.

[26] Centre Saclay and Howard Gutowitz. "Transients, Cycles, and Complexity in Cellular Automata". In: *Physical Review A* 44 (Dec. 1994). DOI: 10.1103/PhysRevA.44.R7881.

[27] Moshe Sipper. "Studying Artificial Life Using a Simple, General Cellular Model". In: *Artificial Life* 2.1 (1994), pp. 1–35.

[28] Alexandre Variengien et al. "Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent". In: *CoRR* abs/2106.15240 (2021). arXiv: 2106.15240. URL: https://arxiv.org/abs/2106.15240.

[29] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[30] Stephen Wolfram. "Computation theory of cellular automata". In: *Communications in Mathematical Physics* 96 (1984), pp. 15–57.

[31] Stephen Wolfram. "Statistical mechanics of cellular automata". In: *Reviews of Modern Physics* 55 (1983), pp. 601–644.

[32] Hector Zenil. "Asymptotic Behavior and Ratios of Complexity in Cellular Automata". In: *International Journal of Bifurcation and Chaos* 23 (Sept. 2013), pp. 50159–. DOI: 10.1142/S0218127413501599.