Exploring the Space of Substitution Systems

Richard Southwell*
Chris Cannings
School of Mathematics and Statistics
University of Sheffield, Hicks Building, Hounsfield Road
Sheffield, S37 RH, United Kingdom
**r.southwell@shef.ac.uk*

Substitution systems, where strings are rewritten according to local rules, have many applications. They are used to model the development of plants, as well as to generate music and architectural designs. Many substitution systems can generate highly complex patterns using only simple rules. This feature can make substitution systems difficult to analyze mathematically. A different approach, pioneered by Stephen Wolfram, is to use computer searches to reveal simple systems with interesting properties. This approach is used to explore a class of systems we call symmetric sequential substitution systems within which a string is repeatedly updated by applying rewrite rules in a non-overlapping way. In this paper several simple examples of these systems are exhibited that produce complex behavior. The dynamics of several of these systems are studied and a system is exhibited that is computationally universal. Applications of symmetric sequential substitution systems are discussed, such as compression and the evaluation of numerical functions.

1. Introduction

The idea behind substitution systems is to repeatedly perform string rewrite operations. These systems have been studied before under a variety of names by different people. Early work focused upon neighborhood independent substitution systems, where each character is replaced in a way that does not dependent upon its surroundings. For example, Thue [1] and Marston Morse [2] both studied the neighborhood independent substitution system with rules $0 \to 01$, $1 \to 10$. Later, Astrid Lindenmayer popularized substitution systems by showing how they can be given geometric interpretations and used to study the development of plants and algae [3, 4].

Lindenmayer also considered more general "neighbor-dependent" substitutions systems, where the way a substring is rewritten depends upon its surroundings. These kinds of systems have subsequently found artistic applications such as the generation of music [5], archi-

https://doi.org/10.25088/ComplexSystems.22.1.1

tecture [6], and computer graphics [7]. Their complexity makes many neighbor-dependent substitution systems difficult to analyze mathematically. One fruitful way to study such complex substitution systems is to take an experimental approach—using computer simulations to get accurate pictures of the dynamics. In [8], Wolfram uses simulations to identify some of the simplest substitution systems that generate complex behavior.

In this paper, we take a similar experimental approach toward a different class of substitution systems. A *symmetric sequential substitution system* is specified by a set $\{a_1 \to b_1, a_2 \to b_2, \ldots, a_n \to b_n\}$ of replacement rules, where $\{a_1, a_2, \ldots, a_n\}$ is a prefix-free set (i.e., no string a_i is a prefix of a string a_j for $i \neq j$). A *replacement rule* $a_i \to b_i$ converts a string a_i to a string b_i. A string is updated under such a system by applying the replacement rules from the left in a non-overlapping way.

In particular, a string x is updated by scanning across it from left to right. Whenever a substring a_i is encountered such that $a_i \to b_i$ is a replacement rule, a_i gets replaced with b_i, and the scan then continues from the immediate right of the newly replaced substring. When the scan reaches the end of the string, the update is complete (see Figure 1).

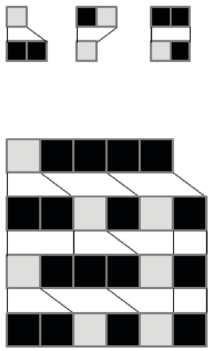

Figure 1. A space-time plot showing the dynamics of the initial string 01111 over the first four time steps under replacement rules $\{0 \to 11, 10 \to 0, 11 \to 01\}$. The initial string is shown at the top and time reads downward. We have used lines to indicate how the strings generate one another; these will be omitted in the subsequent figures.

Using computer searches, we find simple examples of symmetric sequential substitution systems that generate highly complex patterns. We also exhibit a relatively simple symmetric sequential substitution system that is computationally universal, that is, it is capable of simu-

https://doi.org/10.25088/ComplexSystems.22.1.1

lating any computer program. After demonstrating complexity, we show several symmetric sequential substitution systems that can be used to perform arithmetic. We also exhibit a class of systems that are time reversible in a very natural way, and we discuss how these systems can be used for data compression.

2. The Behavior of the Simplest Rules

There are 216 systems of the form $\{0 \to b_1, 10 \to b_2, 11 \to b_3\}$ with b_1, b_2, and b_3 taking values in {0, 1, 00, 01, 10, 11}. This set of simple rules is a good starting point for our exploration.

We examined space-time plots of each of these 216 rules running, with 0 as the initial condition. Several different kinds of behavior were observed. First, 108 of the rules caused the string to reach a fixed point. Also 21 of the rules caused the string to become periodic (the highest period was six). Thirteen of the rules cause the string length to grow linearly with time (creating simple patterns). The remaining 74 rules caused the string's length to grow exponentially. Some of these systems have regularities or nested structures that allow the long-term dynamics to be predicted (e.g., the rule in Figure 2, left; see Theorem 1). However, other systems produce more complex patterns (e.g., the rule in Figure 2, right). The rapid growth rates of the strings make it difficult to say whether the patterns produced by these systems are truly complex, or whether some fractal structure is actually being produced. Let $r\,s$ denote the concatenation of strings r and s, and let r^k denote the string obtained by concatenating together k copies of the string r.

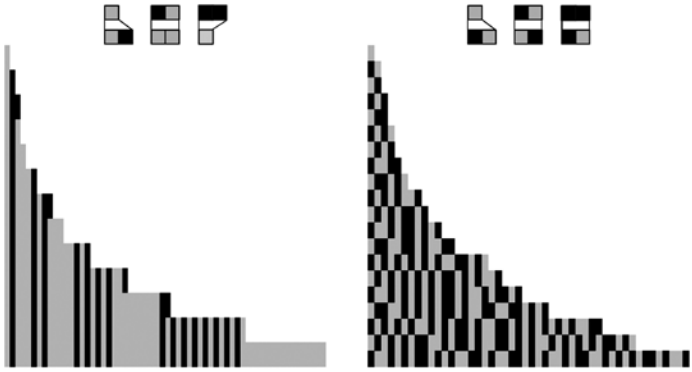

Figure 2. The left rule shows a space-time plot of the system with rules $\{0 \to 01, 10 \to 00, 11 \to 0\}$ evolving over the first 12 updates, starting with the initial condition 0. The right rule shows a space-time plot of the system with rules $\{0 \to 10, 10 \to 01, 11 \to 10\}$ evolving over the first 19 updates.

https://doi.org/10.25088/ComplexSystems.22.1.1

Theorem 1. Let $y(t)$ denote the binary string obtained by evolving a system with rules $\{0 \to 01,\ 10 \to 00,\ 11 \to 0\}$ (Figure 2, left) for t time steps, starting with the initial condition $y(0) = 0$. Now, for each integer $T \geq 3$ we have that:

$$\begin{aligned}
y(4T) &= 01\, a(1)\, b(1)\, a(2)\, b(2) \ldots a(2T-3)\, b(2T-3)\, 0^{m(T)} \\
y(4T+1) &= 01\, a(1)\, b(1)\, a(2)\, b(2) \ldots \\
&\qquad a(2T-3)\, b(2T-3)\, a(2T-2)\, 1\, (01)^{m(T)-2} \\
y(4T+2) &= 01\, a(1)\, b(1)\, a(2)\, b(2) \ldots \\
&\qquad a(2T-2)\, b(2T-2)\, 0^{2m(T)-4}\, 1 \\
y(4T+3) &= 01\, a(1)\, b(1)\, a(2)\, b(2) \ldots \\
&\qquad a(2T-2)\, b(2T-2)\, a(2T-1)\, (10)^{2m(T)-6}\, 11
\end{aligned}$$

where $a(k) = 0^{2^k+1}$, $b(k) = 1\,(01)^{2^k-1}$ and $m(T) = \left(37 \times 4^T + 704\right)/192$. (The proof to Theorem 1 is given in Appendix A.)

We also investigated the dynamics of our 216 rules when the initial condition is the string 0^{100}, consisting of 100 zeros. Under this initial condition, the system with rules $\{0 \to 10,\ 10 \to 01,\ 11 \to 1\}$ produces particularly complex behavior (Figure 3, left). Theorem 2 claims that when the initial condition is an infinite string of zeros, this system produces strings of arbitrary complexity in the sense that, given any binary string β, the system may be evolved for a sufficiently long period of time, the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$ may be applied, and β will be contained in the resulting string.

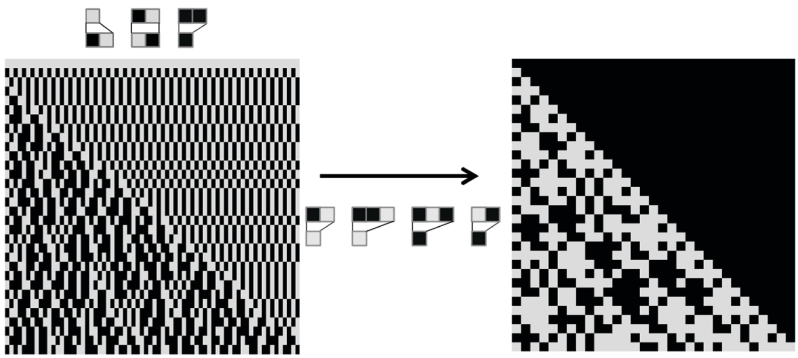

Figure 3. The left shows a space-time plot of the pattern produced by the system with rules $\{0 \to 10,\ 10 \to 01,\ 11 \to 1\}$, starting with a string of zeros (light gray and black represent 0 and 1, respectively). If the substitutions $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$ are applied to each row of the pattern on the left (excluding the top row), then the pattern shown on the right is obtained. Theorem 2 states that if this process is extended for sufficiently many time steps, then every binary string will eventually appear within a row of the pattern on the right.

https://doi.org/10.25088/ComplexSystems.22.1.1

Theorem 2. Let $x(t)$ denote the binary string obtained by evolving a system with rules $\{0 \to 10, 10 \to 01, 11 \to 1\}$ for t time steps, starting with initial condition $x(0) = 0^{\infty}$. Now if β is any finite binary string, then there exists a t such that β is a substring of the string obtained by taking $x(t)$ and then applying the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$. (The proof to Theorem 2 is given in Appendix B.)

3. Systems Involving More than Two Symbols

There are many systems using three symbols with complex behavior (see Figure 4). Some of these produce random-looking patterns, while others produce more complicated patterns that hold a mixture of ordered and disordered regions (much like class 4 cellular automata [9]).

Wolfram's Principle of Computational Equivalence [8] states that whenever the pattern produced by a computation is not obviously simple, that computation will almost always be as complex as any other computation. This principle suggests that some of the complex systems we have already discussed (such as those shown in Figure 4) could be computationally universal (in the sense that they can simulate any single-taped Turing machine). Although we have been unable to prove that any symmetric sequential substitution system with three or less symbols is computationally universal, we have been able to construct a symmetric sequential substitution system with nine symbols that is provably computationally universal (shown in Figure 5).

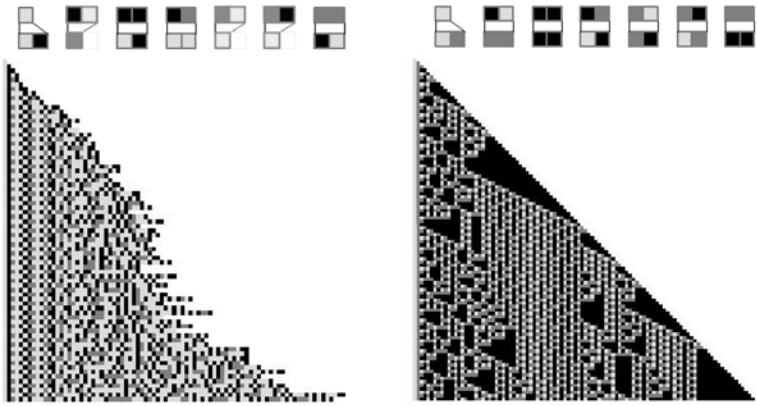

Figure 4. Space-time plots of two systems that grow complex patterns from a single cell. The pattern on the left has a random-looking growth rate. The pattern on the right holds a mixture of ordered and disordered regions.

https://doi.org/10.25088/ComplexSystems.22.1.1

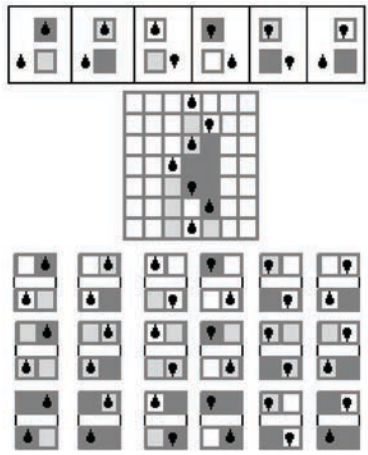

Figure 5. A space-time plot of the simplest known computationally universal Turing machine [10]. The rules are shown at the top. The middle shows the evolution of the system over seven time steps. At the bottom is a set of replacement rules for a symmetric sequential substitution system that emulates this Turing machine directly.

Wolfram's exploration of the simplest Turing machines led to the conjecture [8] that the 2-state, 3-symbol Turing machine shown at the top of Figure 5 is computationally universal. In May 2007, Wolfram offered a $25,000 prize for a proof or disproof of the universality of this Turing machine. In October 2007, Alex Smith won the prize by proving that the Turing machine is universal [10]. Our computationally universal symmetric sequential substitution system was designed to emulate the dynamics of this Turing machine directly.

As Figure 5 depicts, the tape of the Turing machine has three different symbols (represented as white, light gray, and dark gray) and the machine can take two states (represented by an arrow pointing up and an arrow pointing down). The rules at the top of Figure 5 show how this Turing machine rewrites the current symbol, changes state, and moves in response to the current state and tape symbol under the head. To emulate this Turing machine, we introduced nine characters to represent all the possible state-symbol combinations that could be associated with a cell. For example, a white box with an arrow pointing upward inside it represents a cell of the tape with the "white" symbol written upon it, that is, occupied by the head of Turing machine in the "up" state. The symmetric sequential substitution system with nine characters that uses the 18 replacement rules given at the bottom of Figure 5 will emulate the dynamics of the universal Turing machine directly. When the system is updated, only one of the replacement rules will be applied, and this replacement will effectively move the

https://doi.org/10.25088/ComplexSystems.22.1.1

head and alter the tape symbols in accordance with the rules of the universal Turing machine.

4. Performing Multiplication and Other Functions

The way strings change under a symmetric sequential substitution system is similar to the way that cells divide and differentiate in a growing organism. Many organisms are able to regulate their size—growing to a certain limit and then stopping. Many symmetric sequential substitution systems share this property. In fact, some systems can effectively do arithmetic in that they cause a row of n light gray blocks to change into a string of length $f(n)$, which subsequently remains fixed. Figure 6 shows examples with $f(n) = 6\,n$ and $f(n) = 7\,n$.

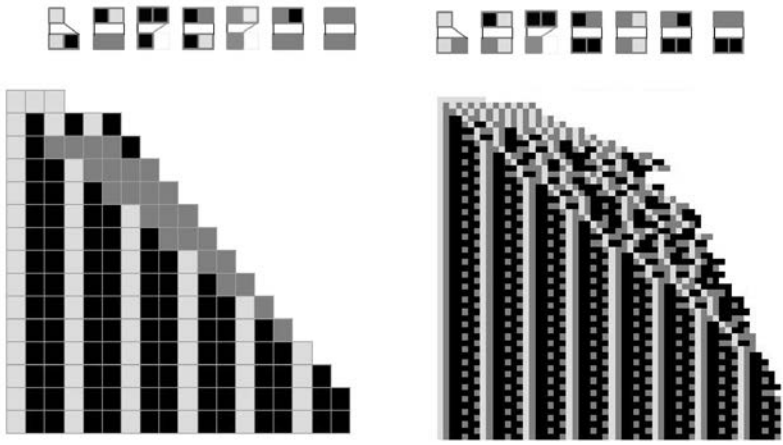

Figure 6. Two systems that effectively perform multiplication. The system on the left causes each string of $n > 1$ light gray blocks to grow into a fixed string of length $6\,n$. The system on the right causes each string of $n > 1$ light gray blocks to grow into a fixed string of length $7\,n$.

The systems shown in Figure 6 were found by doing computer searches through millions of systems. The system at the left in Figure 6 behaves in quite a simple way. The dynamics of the system at the right in Figure 6 are less trivial. This system seems to produce very complex patterns despite accurately evaluating $7\,n$ at least up to $n = 10\,000$. It appears very difficult to prove this system evaluates $7\,n$ for arbitrary n. Table 1 shows systems that can evaluate many other functions. Many of these systems would be very difficult to design, but they can be found quickly by doing computer searches.

https://doi.org/10.25088/ComplexSystems.22.1.1

$f(n)$	Replacement Rules
$3\,n$	$\{0 \to 01,\ 10 \to 10,\ 11 \to 1\}$
$4\,n$	$\{0 \to 02,\ 10 \to 01,\ 11 \to 11,\ 12 \to 11,\ 20 \to 20,$ $21 \to 1,\ 22 \to 11\}$
$5\,n$	$\{0 \to 02,\ 10 \to 10,\ 11 \to 22,\ 12 \to 12,\ 20 \to 10,$ $21 \to 21,\ 22 \to 1\}$
$11\,n$	$\{0 \to 01,\ 10 \to 12,\ 11 \to 2,\ 12 \to 20,\ 20 \to 12,$ $21 \to 01,\ 22 \to \lambda\}$
$n \bmod 3$	$\{0 \to 1,\ 10 \to 20,\ 11 \to 20,\ 12 \to 2,\ 20 \to 01,$ $21 \to \lambda,\ 22 \to \lambda\}$
$\left\lfloor \frac{n}{2} \right\rfloor$	$\{0 \to 1,\ 10 \to 10,\ 11 \to 02,\ 12 \to 1,\ 20 \to 02,$ $21 \to 2,\ 22 \to \lambda\}$

Table 1. The rules behind symmetric sequential substitution systems, which cause each initial string of n zeros to evolve toward a fixed string of length $f(n)$. Note that λ denotes the empty string; $22 \to \lambda$, for example, is equivalent to deleting 22.

5. Reversible Systems

We say that a symmetric sequential substitution system is *completely reversible* upon a set of strings S when the global update function associated with the system is a *one-to-one* mapping from S onto itself. In such a system, each member of S has a unique predecessor. For example, the system shown in Figure 7 is completely reversible upon the set $S = \{0, 1\}^* \times \{2\}$ of all binary strings with a 2 appended. (Here A^* denotes the set of all strings that can be formed from elements of the set A, and $\times$ denotes the Cartesian product.) The unique predecessor of a string x in S can be found by applying the inverse $\{00 \to 0,\ 1 \to 10,\ 01 \to 11,\ 02 \to 12,\ 2 \to 2\}$ of the rule.

A set of replacement rules $R = \{a_1 \to b_1, a_2 \to b_2, \ldots, a_n \to b_n\}$ forms a symmetric sequential substitution system provided $\{a_1, a_2, \ldots, a_n\}$ is a prefix-free set. We define the inverse of R to be $\{b_1 \to a_1, b_2 \to a_2, \ldots, b_n \to a_n\}$. Now suppose that string x is updated under system R to become system y. To guarantee that the inverse of R sends y to x, we need rule R to meet two conditions:

1. The set $\{b_1, b_2, \ldots, b_n\}$ is prefix-free.
2. The string x can be constructed by joining together substrings from $\{a_1, a_2, \ldots, a_n\}$, in some combination.

https://doi.org/10.25088/ComplexSystems.22.1.1

Figure 7. A completely reversible system. Any binary string ending with a 2 can be evolved forward or backward (by inverting the rule) under this system in a reversible way.

Condition 1 is required so that the inverse of R is a well-defined symmetric sequential substitution system. Condition 2 is required so that each part of x is operated upon when it is updated. For example, the rule $R' = \{0 \to 00, 10 \to 1, 11 \to 01\}$ is a symmetric sequential substitution system that satisfies condition 1, but when we update the string $x = 1$ under R' we get $y = 1$, whereas when we update $y = 1$ under the inverse of R' we get $10 \neq x$. This happens because condition 2 is not satisfied.

Consider the set of rules of the form $\{0 \to b_1, 10 \to b_2, 11 \to b_3, 12 \to 02, 2 \to 2\}$ such that $\{b_1, b_2, b_3\} = \{1, 00, 01\}$. Each of these rules satisfies conditions 1 and 2 with respect to each string in $\{0, 1\}^* \times \{2\}$. It follows that each of these six systems is completely reversible upon $\{0, 1\}^* \times \{2\}$ (the system shown in Figure 7 is an example).

There are many completely reversible systems using more symbols (see Figure 8). Some of these rules cause the initial strings to enter high period orbits that include fractal patterns. Other systems cause the initial string to shrink down until a certain point and then expand out again so that the space-time plot is shaped like an hourglass (Figure 8, left). These kinds of rules are interesting because the dynamics often seem to go through a transition from ordered to random-looking behavior at the point of minimal length.

https://doi.org/10.25088/ComplexSystems.22.1.1

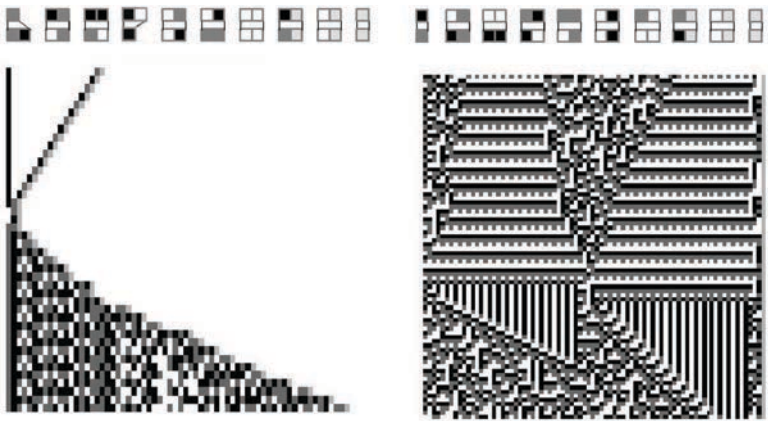

Figure 8. Two systems that are completely reversible for any input in $\{0, 1, 2\}^* \times \{3\}$. The system on the left makes the orderly initial string shrink down to a small size and then blow up into a more complicated pattern. The system on the right preserves the length of the initial string and so is periodic.

The completely reversible systems we have defined make a useful addition to the set of studied models of reversible computation. There are many reasons to study reversible computation. For example, according to Landauer's principle [11], every time an irreversible computation erases a bit of information, there will be an accompanied dissipation of energy in the form of heat. By using reversible computation, this phenomenon can be avoided. Reversible cellular automaton models have become popular models of reversible computation that have found many applications in the modeling of physical systems [12] and in cryptography [13]. Our completely reversible symmetric sequential substitution systems are similar to reversible cellular automata, in the sense that they are reversible models of computation that are based upon local interactions.

6. Using Reversible Systems for Compression

Completely reversible symmetric sequential substitution systems preserve information, and so when such rules reduce the length of the string they are effectively compressing the data [14]. For example, many inputs to the reversible system in Figure 7 induce *hourglass-shaped* space-time plots. It follows that many pieces of data can be compressed by running the system until the string reaches minimal length (at which point we have compressed the string as much as possible under the rule). The compressed data now consists of the resultant string, together with a record of how long the rule has been run

https://doi.org/10.25088/ComplexSystems.22.1.1

for. Given this information, the resultant string may be evolved for the given time under the inverse of the system to recover the original data.

In order to implement this kind of scheme, a *control algorithm* needs to be specified; this is used to decide how many times the replacement rules (or their inverse) should be applied to the string. The purpose of the control algorithm is to attempt to solve the one-dimensional optimization problem that consists of deciding how long to run the system (forward or backward in time) in order to minimize the resultant string's length. A naive control algorithm would consist of evolving the system until the length of the string stops decreasing, and then halting. This type of algorithm would work well for the system shown in Figure 7, but would be inefficient in general. Designing a control algorithm that works in more general cases is extremely challenging.

Results from algorithmic information theory [15] imply fundamental limits upon the performance of this kind of compression scheme. In algorithmic information theory, the *complexity* of a string s is the length of the shortest computer program that generates s (within some prespecified programming language). A string s is said to be algorithmically random when its complexity is equal to its length. A well-known result from algorithmic information theory states that the majority of strings are algorithmically random. This result implies that our compression scheme (for any given completely reversible symmetric sequential substitution system) will be unable to compress the majority of strings.

Another result from algorithmic information theory (Chaitin's incompleteness theorem [15]) implies that the complexity of most strings is uncomputable. This implies that in general, there can be no finite control algorithm that always applies the replacement rules the correct number of times so as to minimize the length of the output. A concrete example of a completely reversible symmetric sequential substitution system for which it is difficult to find an appropriate control algorithm is shown in Figure 9. In complex scenarios like this, Wolfram's Principle of Computational Equivalence [8] implies that there may be no way to shortcut the evolution of the system and quickly determine when the length of the string will be minimized.

We wrote a compression procedure based upon completely reversible symmetric sequential substitution systems. Since a given reversible system can only exploit certain kinds of regularities in the data, our procedure tests the abilities of many completely reversible symmetric sequential substitution system to compress the data, and then outputs the rule that performs best, how long it should be run for, and the form of the resulting string. When we "test" a given sys-

https://doi.org/10.25088/ComplexSystems.22.1.1

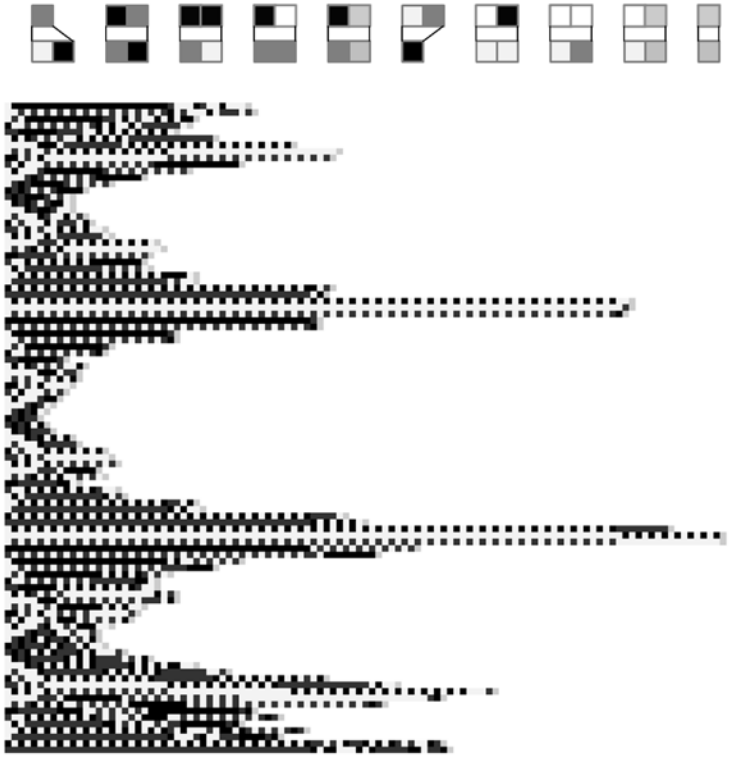

Figure 9. A space-time plot of a system that is completely reversible upon $\{0, 1, 2\}^* \times \{3\}$, where it is difficult to determine how long the system should be run in order to minimize the length of the string. The initial condition is 211111111111111111111111110222102212223. After 50 updates, the string is compressed to 0101213. If the system continues to evolve for 50 more time steps, then the fluctuations in the string length become increasingly rapid. For example, at time step 150 the string length reaches 1062, at time step 175 the string length has gone back down to 558, and at time step 250 the string has length 457 275.

tem, we just evolve the string under the system until its length stops decreasing; at this point we halt and examine the length of the resultant string. (We use this kind of naive control algorithm because [as we have already discussed] it is fundamentally difficult to design an optimal control algorithm.) In order to achieve performance levels similar to those of popular compression methods such as the Lempel–Ziv–Welch [16] algorithm, we need to use several thousand reversible symmetric sequential substitution systems within our procedure. This makes our procedure too slow for many practical scenarios. Moreover, our procedure can only exploit local patterns within the data,

https://doi.org/10.25088/ComplexSystems.22.1.1

whereas schemes like Huffman coding [14] are capable of exploiting non-local patterns.

However, despite its shortcomings there are some areas where our completely reversible symmetric sequential substitution system-based scheme has definite applications. When a compression scheme is successful, it reveals regularities within the data. This means that compression schemes can be used for data analysis. For example, in [17] compression-based clustering is used to reveal patterns in genetics, languages, and astrophysics. Also in [18], Hector Zenil uses a compression-based method to classify the behavior of cellular automata. To quote Gregory Chaitin [19], "understanding is compression." When a simple completely reversible symmetric sequential substitution system is found to *significantly* compress the data, it can be thought of as a model for that data. Such a model, based upon symmetric sequential substitution system rules, is attractive because of its simple nature. Also, the dynamic nature of our compression scheme makes it useful for analysis. For example, if the same completely reversible symmetric sequential substitution system is found to compress many data strings, then you may wish to run the substitution system for different amounts of time, to extrapolate from the dataset.

7. Conclusion

Our exploration of the space of symmetric sequential substitution systems has revealed a diverse range of different behavior. We have discussed applications of these systems such as evaluation of arithmetic operations and data compression. Our findings are further evidence that many useful computations can be performed using extremely simple programs.

One interesting direction for future research is to further study the dynamics of completely reversible systems and investigate what kinds of behavior and growth rates completely reversible systems can exhibit. Also, compression schemes based upon completely reversible systems raise some interesting questions for algorithmic information theory, because these systems establish a complexity measure similar to Kolmogorov complexity.

Appendix

A. Proof of Theorem 1

Let $\Phi(s)$ denote the string obtained by updating string s under the replacement rules $\{0 \to 01, 10 \to 00, 11 \to 0\}$. Now we have $y(0) = 0$ and $y(t) = \Phi(y\,(t-1))$, $\forall\ t \geq 1$. We say that a binary string $s \in \{0, 1\}^*$

https://doi.org/10.25088/ComplexSystems.22.1.1

is *clean* when it can be constructed by joining together substrings from the set {0, 10, 11}, in some combination. The form of a clean prefix has no effect on the way the remainder of a string gets updated. In other words, if s is clean, then $\Phi(sr) = \Phi(s)\,\Phi(r)$.

In our case, we have that $01\,a(1) = 010^{2^1+1} = 01\,000 = (0)\,(10)\,(0)\,(0)$ is clean. Moreover, updating this string yields

$$\Phi(01\,a\,(1)) = \Phi(01\,000) = 01\,000\,101 = 01\,a(1)\,b(1). \tag{A.1}$$

Also, $\forall\ k \geq 1$ we have that the string $b\,(k)\,a\,(k+1)$ is clean. Updating this string yields

$$\begin{aligned}\Phi(b(k)\,a(k+1)) &= \Phi\left((10)^{2^k-1}\,10^{2^{k+1}+1}\right) = \\ &\Phi\left((10)^{2^k}\,0^{2^{k+1}}\right) = (00)^{2^k}\,(01)^{2^{k+1}} = \\ &0^{2^{k+1}+1}\,(10)^{2^{k+1}-1}\,1 = a(k+1)\,b(k+1).\end{aligned} \tag{A.2}$$

Also, $\forall\ i \geq 2\ \forall\ k \geq 1$ we have that the string $b(k)\,0^i$ is clean, and updating $b(k)\,0^i$ yields

$$\begin{aligned}&\Phi\left(b(k)\,0^i\right) = \Phi\left((10)^{2^k-1}\,10^i\right) = \Phi\left((10)^{2^k}\,0^{i-1}\right) = \\ &(00)^{2^k}\,(01)^{i-1} = 0^{2^k+1}\,1\,(01)^{i-2} = a(k+1)\,1\,(01)^{i-2}.\end{aligned} \tag{A.3}$$

We can verify that our result gives the value of $y(12)$ when $T = 3$ correctly by computing $y(12)$ directly (also note that $y(12)$ corresponds to the bottom row of the space-time plot shown in Figure 2, left). Direct computation yields

$$\begin{aligned}&y(12) = \\ &\quad 010\,001\,010\,000\,010\,101\,010\,000\,000\,001\,010\,101\,\ddots \\ &\qquad 010\,101\,010\,000\,000\,000\,000\,000 = \\ &\quad y(4\times 3) = 01\,a(1)\,b(1)\,a(2)\,b(2)\,a(3)\,b(3)\,0^{16}\end{aligned} \tag{A.4}$$

(where $m(3) = 16$), as required.

So the equation for $y(4\,T)$ where $T = 3$ holds. Now, we shall use induction. For $T \geq 3$, suppose that our result holds true for $y(4\,T)$ in the sense that $y(4\,T) = 01\,a(1)\,b(1)\,a(2)\,b(2)\,..\,a(2\,T-3)\,b(2\,T-3)\cdot 0^{m\,(T)}$. Now in this case we have that

$$\begin{aligned}y(4\,T+1) &= \Phi(y\,(4\,T)) = \Phi(01\,a(1))\,\Phi(b(1)\,a(2))\,\Phi(b(2)\,a(3))\,.. \\ &\quad \Phi(b(2\,T-4)\,a(2\,T-3))\,\Phi\left(b(2\,T-3)\,0^{m\,(T)}\right) = \\ &\quad [01\,a(1)\,b(1)][a(2)\,b(2)][a(3)\,b(3)]\,.. \\ &[a(2\,T-3)\,b(2\,T-3)]\left[a(2\,T-2)\,1\,(01)^{m\,(T)-2}\right] = \\ &\quad 01\,a(1)\,b(1)\,a(2)\,b(2)\,a(3)\,b(3)\,.. \\ &\quad a(2\,T-3)\,b(2\,T-3)\,a(2\,T-2)\,1\,(01)^{m\,(T)-2}\end{aligned}$$

https://doi.org/10.25088/ComplexSystems.22.1.1

and so our result holds when $y(4\,T+1)$. Note that here we make use of equations (A.1), (A.2), and (A.3) to go from the second line to the third line. Given that our result holds true for $y(4\,T+1)$, we have

$$\begin{aligned} y(4\,T+2) &= \Phi(01\,a(1))\,\Phi(b(1)\,a(2))\,\Phi(b(2)\,a(3))\,..\\ &\quad \Phi(b(2\,T-3)\,a(2\,T-2))\,\Phi\big(1\,(01)^{m\,(T)-2}\big) =\\ &\quad 01\,a(1)\,b(1)\,a(2)\,b(2)\,a(3)\,b(3)\,..\\ &\quad a(2\,T-2)\,b(2\,T-2)\,(0)^{2\,m\,(T)-4}\,1 \end{aligned}$$

and so our result holds when $y(4\,T+2)$. Given that our result holds true for $y(4\,T+2)$, we have

$$\begin{aligned} &y\,(4\,T+3) = \Phi\,(01\,a\,(1))\,\Phi\,(b\,(1)\,a\,(2))\\ &\Phi\,(b\,(2)\,a\,(3))\,..\,\Phi\,(b\,(2\,T-3)\,a\,(2\,T-2))\,\Phi\\ &\big(b\,(2\,T-2)\,(0)^{2\,m\,(T)-4}\,1\big) = 01\,a\,(1)\,b\,(1)\,a\,(2)\,b\,(2)\,a\,(3)\\ &b\,(3)\,..\,a\,(2\,T-2)\,b\,(2\,T-2)\,a\,(2\,T-1)\,(10)^{2\,m\,(T)-6}\,11 \end{aligned}$$

and so our result holds when $y(4\,T+3)$. Given that our result holds true for $y\,(4\,T+3)$, we have

$$\begin{aligned} y(4\,T+4) &= \Phi(01\,a(1))\,\Phi(b(1)\,a(2))\,\Phi(b(2)\,a(3))\,..\\ &\quad \Phi(b(2\,T-2)\,a(2\,T-1))\,\Phi\big((10)^{2\,m\,(T)-6}\,11\big) =\\ &\quad [01\,a(1)\,b(1)][a(2)\,b(2)][a(3)\,b(3)]\,..\\ &[a(2\,T-1)\,b(2\,T-1)]\,0^{4\,m(T)-11} =\\ &\quad 01\,a(1)\,b(1)\,a(2)\,b(2)\,a(3)\,b(3)\,..\\ &\quad a(2\,T-1)\,b(2\,T-1)\,0^{m(T+1)} \end{aligned}$$

and so our result holds when $y(4\,T+4) = y(4\,(T+1))$. Here we get from the second line to the third line using the fact that

$$m(T) = \frac{37\times 4^{T} + 704}{192},$$

and so $4\,m(T) - 11 = m(T+1)$.

We have shown that the result holds for $y(4\,T) : T = 3$. We have also shown that, if $T \geq 3$ is such that the result holds for $y(4\,T)$, then the result holds for $y(4\,T+1)$, $y(4\,T+2)$, $y(4\,T+3)$, and $y(4\,(T+1))$. Our theorem therefore follows by induction with T. □

B. Proof of Theorem 2

During this proof we shall refer to the symmetric sequential substitution system with replacement rules $\{0 \to 10,\ 10 \to 01,\ 11 \to 1\}$ as

https://doi.org/10.25088/ComplexSystems.22.1.1

system 58 because this is the 58th rule that appears in our enumeration of the rule set.

Let $\psi(s)$ denote the binary string obtained by applying the replacement rules $\{0 \to 10, 10 \to 01, 11 \to 1\}$ to binary string s. Now we shall have that $x(0) = 0^\infty$, and $x(t) = \psi(x(t-1))$ will be the string obtained by evolving system 58 for $t \geq 1$ time steps.

Now we shall define a one-dimensional cellular automaton with cells indexed with positive integers. The cells take values in $\{1, 2, 3, 4\}$. We let

$$Y^t = Y_1^t, Y_2^t, Y_3^t, \ldots \in \{1, 2, 3, 4\}^\infty$$

denote the state of the cellular automaton at time $t \geq 0$. The initial state is

$$Y^0 = 14^\infty = 1, 4, 4, 4, \ldots.$$

The value of cell $i \geq 1$ may be determined at time $t \geq 1$ using the formula

$$Y_i^t = \begin{cases} f\left(1, Y_i^{t-1}\right) & \text{if } i = 1 \\ f\left(Y_{i-1}^{t-1}, Y_i^{t-1}\right) & \text{otherwise,} \end{cases}$$

where the mapping $f : \{1, 2, 3, 4\}^2 \mapsto \{1, 2, 3, 4\}$ is given by the table:

		b			
	$f(a, b)$	1	2	3	4
	1	4	2	4	1
a	2	4	2	4	1
	3	2	3	2	4
	4	2	3	2	4

From now on we shall refer to this cellular automaton as the *f-cellular automaton*. Let us define the mapping $\gamma : \{1, 2, 3, 4\} \mapsto \{0, 1\}^*$ such that $\gamma(1) = 10$, $\gamma(2) = 110$, $\gamma(3) = 101$, $\gamma(4) = 01$. We will use γ to convert system 58 into the f-cellular automaton. The way this works is pictured in Figure B1. Lemma 1 states that the string in system 58 present at time step $t + 3$ can be obtained by taking the state of the f-cellular automaton at time t and making the substitutions $1 \to 10$, $2 \to 110$, $3 \to 101$, $4 \to 01$.

Lemma 1. For each $t \geq 0$ we have that

$$x(t+3) = \gamma\left(Y_1^t\right)\gamma\left(Y_2^t\right)\gamma\left(Y_3^t\right)\ldots.$$

https://doi.org/10.25088/ComplexSystems.22.1.1

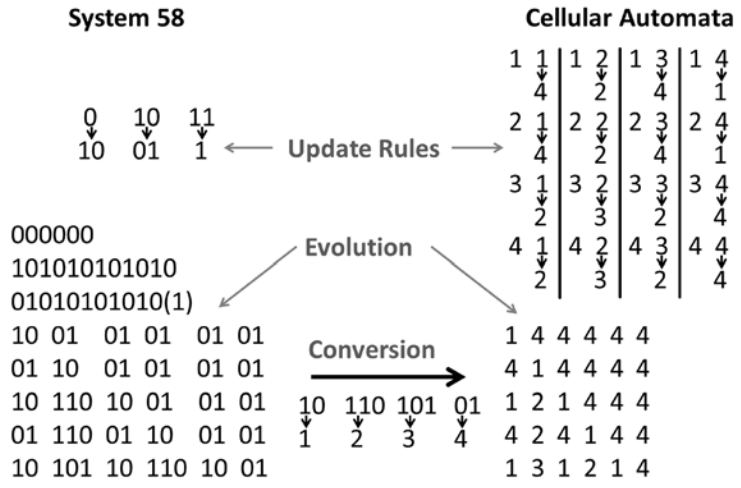

Figure B1. An illustration of the relationship between system 58 and the f-cellular automaton. On the left, we show the dynamics of system 58 over the first seven updates, starting with an initial string of zeros (our initial condition only holds six zeros in this finite example). At the bottom right of the figure, we show part of the space-time plot of the f-automaton. Successive rows, reading downward, correspond to states Y^0, Y^1, .., Y^4 on successive time steps. The update function $f(a, b)$ behind this cellular automaton is illustrated at the top right. In the center, we illustrate the replacement rules that convert strings from system 58 (generated after three or more updates) to rows of the space-time plot of the f-cellular automaton. (To do our conversion, we ignore the bracketed (1) at the end of the third row. This problem does not occur when 0^∞ is our initial condition.)

Proof. For a binary string $w \in \{0, 1\}^*$ of length $|w| > 1$, let us define

$$w^- = \begin{cases} w & \text{if } w_{|w|} = 0 \\ w_1\, w_2\, ..\, w_{|w|-1} & \text{if } w_{|w|} = 1 \end{cases}$$

to be equal to the string obtained by taking w and deleting its last character, if it is a zero. Also, let us define

$$w^e = \begin{cases} \lambda & \text{if } w_{|w|} = 0 \\ 1 & \text{if } w_{|w|} = 1 \end{cases}$$

to be equal to the empty string λ when w ends in 0, and equal to 1 when w ends in 1. Now clearly, we will have that w is always equal to the concatenation $w = w^-\, w^e$.

https://doi.org/10.25088/ComplexSystems.22.1.1

Let us first consider the case where $t = 0$. In this case the states of system 58 on the first three time steps are:

$$x(0) = 0^{\infty}$$
$$x(1) = (10)^{\infty}$$
$$x(2) = (01)^{\infty} = 0\,(10)^{\infty}$$
$$x(3) = 10\,(01)^{\infty}.$$

When $t = 0$ we have $x(t+3) = 10\,(01)^{\infty} = \gamma(1)[\gamma(4)]^{\infty} = \gamma\big(Y_1^0\big)\gamma\big(Y_2^0\big)\gamma\big(Y_3^0\big)\ldots$, and so our result holds true when $t = 0$. Now we shall use proof by induction.

Suppose that our result holds for some $t \geq 0$, in the sense that

$$x(t+3) = \gamma\big(Y_1^t\big)\gamma\big(Y_2^t\big)\gamma\big(Y_3^t\big)\ldots \tag{B.1}$$

Now since $\gamma\big(Y_i^t\big) = \big[\gamma\big(Y_i^t\big)\big]^{-}\big[\gamma\big(Y_i^t\big)\big]^{e}$, for each i, we shall have that

$$x(t+3) = \big[\gamma\big(Y_1^t\big)\big]^{-}\big[\gamma\big(Y_1^t\big)\big]^{e}\big[\gamma\big(Y_2^t\big)\big]^{-}\big[\gamma\big(Y_2^t\big)\big]^{e}\big[\gamma\big(Y_3^t\big)\big]^{-}\big[\gamma\big(Y_3^t\big)\big]^{e}\ldots$$

Now $\forall\, i \geq 2$ let us define the string $\nu(i) \in \{0, 1\}^{*}$ such that $\nu(i) = \big[\gamma\big(Y_{i-1}^t\big)\big]^{e}\big[\gamma\big(Y_i^t\big)\big]^{-}$. Also, let us define $\nu(1) = \big[\gamma\big(Y_1^t\big)\big]^{-}$. Now we have that

$$x(t+3) = \nu(1)\,\nu(2)\,\nu(3)\ldots$$

is equal to the concatenation of all $\nu(i)$.

The important point is that $\forall\, i$ we have $\gamma\big(Y_i^t\big) \in \{10, 110, 101, 01\}$, and so $\big[\gamma\big(Y_i^t\big)\big]^{-}$ has its last character equal to zero.

Now $x^{t+4}(0^{\infty}) = \psi(x\,(t+3))$ is equal to the string obtained by applying the ψ update operator to $x(t+3)$. Now since

$$x(t+3) = \nu(1)\,\nu(2)\,\nu(3)\ldots$$

can be partitioned into parts $\nu(i)$ that each end with a zero, we have that

$$x(t+4) = \psi(\nu(1)\,\nu\,(2)\,\nu(3)\ldots) = \psi(\nu(1))\,\psi(\nu(2))\,\psi(\nu(3))\ldots. \tag{B.2}$$

This is true because each string $\nu(i)$ can be partitioned into substrings of the form 0, 10, 11, and so the way $\nu(i)$ gets updated has no effect upon the way that $\nu(i+1)$ gets updated.

Now we will show that $\gamma\big(Y_i^{t+1}\big) = \psi(\nu(i))$ for each $i \geq 1$. We will do this by considering the different forms that $\nu(i)$ can take.

Let us begin by considering the case where $i \geq 2$. In this case, we have $\nu(i) = \big[\gamma\big(Y_{i-1}^t\big)\big]^{e}\big[\gamma\big(Y_i^t\big)\big]^{-}$, where

$$\gamma\big(Y_{i-1}^t\big),\ \gamma\big(Y_i^t\big) \in \{10, 110, 101, 01\}.$$

https://doi.org/10.25088/ComplexSystems.22.1.1

First, let us consider the case where $\gamma\left(Y_{i-1}^t\right) \in \{101, 01\}$ (in other words, let us consider the case where $Y_{i-1}^t \in \{3, 4\}$). In this case $\left[\gamma\left(Y_{i-1}^t\right)\right]^e = 1$. Now we shall verify that, for each possible value of $Y_i^t \in \{1, 2, 3, 4\}$, we have that $\gamma\left(Y_i^{t+1}\right) = \psi(\upsilon(i))$:

1. If $Y_i^t = 1$ then $\gamma(Y_i^t) = 10 = [\gamma(Y_i^t)]^-$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 110$ and so $\psi(\upsilon(i)) = 110$. Also, since
   $Y_{i-1}^t \in \{3, 4\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(3, 1) = f(4, 1) = 2$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(2) = 110$ as required.
2. If $Y_i^t = 2$ then $\gamma(Y_i^t) = 110 = [\gamma(Y_i^t)]^-$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 1110$ and so $\psi(\upsilon(i)) = 101$. Also, since
   $Y_{i-1}^t \in \{3, 4\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(3, 2) = f(4, 2) = 3$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(3) = 101$ as required.
3. If $Y_i^t = 3$ then $\gamma(Y_i^t) = 101$ and $[\gamma(Y_i^t)]^- = 10$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 110$ and so $\psi(\upsilon(i)) = 110$. Also, since
   $Y_{i-1}^t \in \{3, 4\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(3, 3) = f(4, 3) = 2$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(2) = 110$ as required.
4. If $Y_i^t = 4$ then $\gamma(Y_i^t) = 01$ and $[\gamma(Y_i^t)]^- = 0$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 10$ and so $\psi(\upsilon(i)) = 01$. Also, since
   $Y_{i-1}^t \in \{3, 4\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(3, 4) = f(4, 4) = 4$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(4) = 01$ as required.

So we have proved that the result holds whenever $i \geq 2$ and $Y_{i-1}^t \in \{3, 4\}$. Now let us consider the case where $i \geq 2$ and $Y_{i-1}^t \in \{1, 2\}$. Once again we shall verify that, for each possible value of $Y_i^t \in \{1, 2, 3, 4\}$, we have $\gamma\left(Y_i^{t+1}\right) = \psi(\upsilon(i))$:

1. If $Y_i^t = 1$ then $\gamma(Y_i^t) = 10 = [\gamma(Y_i^t)]^-$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma((Y_i^t)]^- = 10$ and so $\psi(\upsilon(i)) = 01$. Also, since
   $Y_{i-1}^t \in \{1, 2\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(1, 1) = f(2, 1) = 4$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(4) = 01$ as required.

https://doi.org/10.25088/ComplexSystems.22.1.1

2. If $Y_i^t = 2$ then $\gamma(Y_i^t) = 110 = [\gamma(Y_i^t)]^-$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e\,[\gamma\,(Y_i^t)]^- = 110$ and so $\psi(\upsilon(i)) = 110$. Also, since
   $Y_{i-1}^t \in \{1, 2\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(1, 2) = f(2, 2) = 2$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(2) = 110$ as required.
3. If $Y_i^t = 3$ then $\gamma(Y_i^t) = 101$ and $[\gamma(Y_i^t)]^- = 10$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 10$ and so $\psi(\upsilon(i)) = 01$. Also, since
   $Y_{i-1}^t \in \{1, 2\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(1, 3) = f(2, 3) = 4$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(4) = 01$ as required.
4. If $Y_i^t = 4$ then $\gamma(Y_i^t) = 01$ and $[\gamma(Y_i^t)]^- = 0$ and so
   $\upsilon(i) = [\gamma(Y_{i-1}^t)]^e[\gamma(Y_i^t)]^- = 0$ and so $\psi(\upsilon(i)) = 10$. Also, since
   $Y_{i-1}^t \in \{1, 2\}$ we shall have that
   $Y_i^{t+1} = f(Y_{i-1}^t, Y_i^t) = f(1, 4) = f(2, 4) = 1$, and so
   $\gamma\left(Y_i^{t+1}\right) = \gamma(1) = 10$ as required.

So we have proved that the result holds whenever $i \geq 2$. The proof for the case with $i = 1$ is the same, because when $i = 1$ we have $\forall\ Y_i^t \in \{1, 2, 3, 4\}$ that $Y_i^{t+1} = f\left(1, Y_i^t\right)$ and $\upsilon(i) = \left[\gamma\left(Y_i^t\right)\right]^-$.

Hence we have shown that, when equation (B.1) holds true for some t, we shall have that $\gamma\left(Y_i^{t+1}\right) = \psi(\upsilon(i))$ is true $\forall\ i \geq 1$, and so, by equation (B.2), we have that equation (B.1) will also be true for $t + 1$. Now since equation (B.1) is true when $t = 0$, we can use induction with t to show that equation (B.1) is true for every $t \geq 0$. □

An equivalent way to state Lemma 1 is to say that the string $x(t + 3)$ from system 58 can be converted into the state Y^t of the f-cellular automaton by applying the replacement rules $10 \to 1$, $110 \to 2$, $101 \to 3$, $01 \to 4$ (in a non-overlapping way). Our next main objective will be to show that, given any binary string β, a state Y^T may be found of the f-cellular automaton, such that β is contained within the binary string obtained by applying the replacement rules $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$ to Y^T. (Note that making these replacements has the same effect as applying the function $\left\lfloor \frac{\cdot}{3} \right\rfloor$.) Once we have shown this, we have effectively proved our theorem because we have that $x(T + 3)$ can be converted into Y^T by applying the replacement rules $10 \to 1$, $110 \to 2$, $101 \to 3$, $01 \to 4$. It follows that, if $x(T + 3)$ is taken and the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$ are applied then the resulting binary string will contain β as a sub-

https://doi.org/10.25088/ComplexSystems.22.1.1

string. (This is because applying the replacement rules $10 \to 1$, $110 \to 2$, $101 \to 3$, $01 \to 4$ followed by the replacement rules $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$ has the same effect as applying the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$.)

B.1 Analysis of the Diagonals

Consider the space-time plot of the f-cellular automaton. The top row of this space-time plot is the initial condition of the cellular automaton pattern, which is $Y^0 = Y^0_1\, Y^0_2\, Y^0_3 \ldots = 14^\infty$. The subsequent rows Y^1, Y^2 .. of this space-time plot show the states of the cellular automaton on the subsequent time steps.

For each $n \neq 1$ let us define the n^{th} diagonal $D(n) = D(n)_1\, D(n)_2\, D(n)_3 \ldots$ to be such that for each $i \geq 1$, we have $D(n)_i = Y^{n+i-2}_i$. Essentially $D(n)$ is the infinite sequence of characters from the space-time plot that is encountered by starting at cell Y^{n-1}_1 and moving diagonally downward and to the right (see Figure B2).

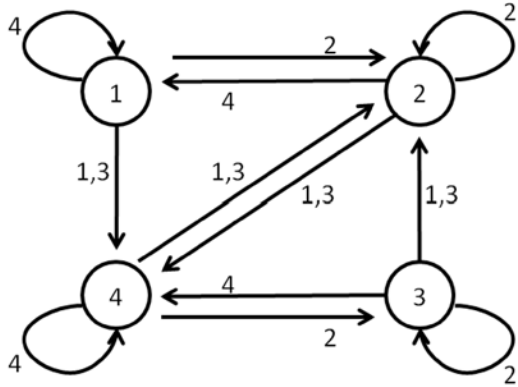

Figure B2. An illustration of the mapping $G_b(a)$ that can be used to determine diagonals from previous diagonals. The vertices represent characters in {1, 2, 3, 4}; the edges are also labeled with characters in {1, 2, 3, 4}. Note that the edges with one end in {1, 2} and the other end in {3, 4} are labeled with both 1 and 3. It can be shown that $D(n+1)_i$ is the vertex of this graph arrived at by starting at vertex $D(n+1)_1$ and then traversing edges with labels $D(n)_2$, $D(n)_3$, .., $D(n)_i$ in sequence.

Now we shall describe how $D(n+1)$ can be computed from $D(n)$. In particular, we have

$$\begin{aligned} D(n+1)_{i+1} &= Y^{n+i}_{i+1} = f\left(Y^{n+i-1}_i,\, Y^{n+i-1}_{i+1}\right) = \\ & f(D(n+1)_i,\, D(n)_{i+1}) = G_{D(n)_{i+1}}(D(n+1)_i) \end{aligned} \tag{B.3}$$

https://doi.org/10.25088/ComplexSystems.22.1.1

where we define $G_b(a) := f(a, b)$, $\forall\, a, b \in \{1, 2, 3, 4\}$. Here $G_b(a)$ is the vertex of the network shown in Figure B2, which is arrived at by starting at vertex a and then traveling along an outwardly pointing edge with label b.

For an element $a \in \{1, 2, 3, 4\}$, we use the notation $\left\lfloor \frac{a}{3} \right\rfloor$ as shorthand to denote the result of applying the replacement rules $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$ to a. This makes sense because

$$\left\lfloor \frac{a}{3} \right\rfloor = \begin{cases} 0 & \text{if } a \in \{1, 2\} \\ 1 & \text{if } a \in \{3, 4\}. \end{cases}$$

In Lemma 2 we shall describe how the diagonals $D(n)$ are periodic sequences with special properties (see Figure B3).

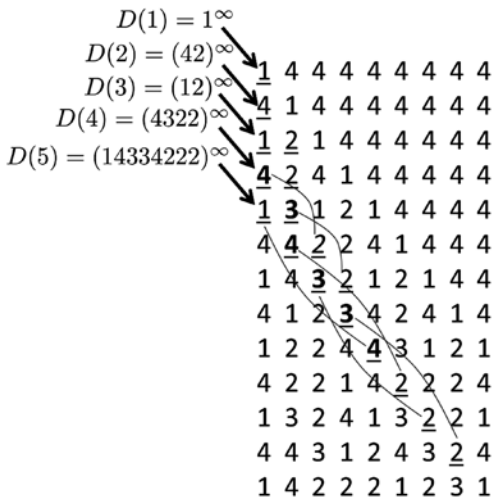

Figure B3. An illustration of how Lemma 2 describes the diagonals of the space-time plot of the f-cellular automaton. The underlined characters show the repeating parts of the diagonals. Note that the fourth diagonal onward has the property that $\left\lfloor \frac{D(n)_{i+2^{n-3}}}{3} \right\rfloor = 1 - \left\lfloor \frac{D(n)_i}{3} \right\rfloor$. We have illustrated this using gray lines to pair up entries that get mapped to different values under the $\left\lfloor \frac{-}{3} \right\rfloor$ operation.

Lemma 2. For each $n \geq 4$ we have that the n^{th} diagonal $D(n)$ of the f-cellular automata space-time plot satisfies the following three conditions:

1. For each $i \geq 1$ we have $D(n)_i = D(n)_{i+2\,h(n)}$, where $h(n) = 2^{n-3}$ is equal to half of the period of $D(n)$.

https://doi.org/10.25088/ComplexSystems.22.1.1

2. For each $i \geq 1$ we have $\left\lfloor \frac{D(n)_i}{3} \right\rfloor = 1 - \left\lfloor \frac{D(n)_{i+h(n)}}{3} \right\rfloor$.

3. There is an odd number of indices $i \in \{1, 2, .., 2\,h(n)\}$ such that $D(n)_i \in \{1, 3\}$ and there is an odd number of indices $i \in \{1, 2, .., 2\,h(n)\}$ such that $D(n)_i \in \{2, 4\}$. (Essentially this condition says that the repeating part of $D(n)$ holds an odd number of values in $\{1, 3\}$ and an odd number of values in $\{2, 4\}$.)

Proof. We start by showing that the lemma holds true when $n = 4$. First, note that $D(1) = 1^{\infty}$. This can easily be seen by examining Figure B2 and noting that $\forall\, i \geq 1$ we have that the i^{th} column, $Y_i^0, Y_i^1, Y_i^2, \ldots$ of the space-time plot has prefix $4^{i-1}\,1$. (This can be proved by induction using the facts that $\left[Y_i^t = 4, Y_{i+1}^t = 4\right] \Rightarrow Y_{i+1}^{t+1} = 4$ and $\left[Y_i^t = 1, Y_{i+1}^t = 4\right] \Rightarrow Y_{i+1}^{t+1} = 1$.)

Now that we have $D(1) = D(1)_1\, D(1)_2\, D(1)_3 \ldots = Y_1^0\, Y_2^1\, Y_3^2 \ldots = 1^{\infty}$, we can go on to compute the next diagonals. The first column of the space-time plot has the form $Y_1^0\, Y_1^1\, Y_1^2 \ldots = (14)^{\infty} = D(1)_1\, D(2)_1\, D(3)_1 \ldots$ and so we have that if n is odd then $D(n)_1 = 1$ and if n is even then $D(n)_1 = 4$.

Now let us compute $D(2)$. We know $D(2)_1 = 4$, and from equation (B.3) we have that $D(2)_2 = G_{D(1)_2}(D(2)_1) = G_1(4) = 2$. Also we get $D(2)_3 = G_{D(1)_3}(D(2)_2) = G_1(2) = 4$. By continuing in this way it can easily be seen that $D(2) = (42)^{\infty}$. Now let us compute $D(3)$. First, $D(3)_1 = 1$. Now $D(3)_2 = G_{D(2)_2}(D(3)_1) = G_2(1) = 2$. Also $D(3)_3 = G_{D(2)_3}(D(3)_2) = G_4(2) = 1$. Clearly the subsequent characters of $D(3)$ will be specified by $D(2)$ in a similar way, and since $D(2)$ has period two we shall have that $D(3) = (12)^{\infty}$. In a similar way, the value of $D(3)$ may be used to show that $D(4) = (4322)^{\infty}$.

Now we can see that $n = 4$ satisfies the conditions of our lemma. To see that $D(4)$ satisfies condition 1, note that $D(4) = (4322)^{\infty}$ has a period of $4 = 2\,h(4)$. To see that $D(4)$ satisfies condition 2, note that $D(4)_1, D(4)_2 \in \{3, 4\}$ and $D(4)_3, D(4)_4 \in \{1, 2\}$. To see that $D(4)$ satisfies condition 3, note that the repeating part, 4322, of $D(4)$ contains an odd number of entries in $\{1, 3\}$ and an odd number of entries in $\{2, 4\}$.

So now that we have established that the result holds for $n = 4$, we can continue with the proof by induction. Suppose that our lemma holds true for some $n \geq 4$. We shall prove that our lemma also holds true for $n + 1$.

We shall start by showing that $D(n + 1)$ satisfies condition 2.

https://doi.org/10.25088/ComplexSystems.22.1.1

We are assuming that $D(n)$ satisfies condition 1. This means we are assuming that $2^{n-2} = 2\,h(n) = h(n+1)$ is equal to the period of $D(n)$. It follows that $D(n)$ is of the form $D(n) = r^{\infty}$ where $r \in \{1, 2, 3, 4\}^{2\,h(n)}$ is the repeating part of $D(n)$. Now, since we are also supposing that $D(n)$ satisfies condition 3, we have that the sequence $D(n)_1\, D(n)_2\, ..\, D(n)_{2\,h(n)}$ (which is equal to $r_1\, r_2\, ..\, r_{2\,h(n)}$) has an odd number of entries in $\{1, 3\}$ and an odd number of entries in $\{2, 4\}$.

Recall that $D(n+1)_{i+1} = G_{D(n)_{i+1}}(D(n+1)_i)$. Now, by using this fact repeatedly we can get that

$$\begin{aligned} D(n+1)_{i+h(n+1)} &= G_{D(n)_{i+h(n+1)}} \circ G_{D(n)_{i+h(n+1)-1}} \circ .. \\ &\quad \circ G_{D(n)_{i+1}}(D(n+1)_i), \end{aligned} \tag{B.4}$$

where $\circ$ denotes functional composition.

Recall that $G_e(v)$ is the vertex of the digraph in Figure B2, which is arrived at by starting at vertex v, and then moving along an edge with label e. Now this implies that equation (B.4) may be interpreted as describing a graph walk. In particular, equation (B.4) says that $D(n+1)_{i+h(n+1)}$ is the vertex which is arrived at by starting from vertex $D(n+1)_i$ (within the graph shown in Figure B2), and then traveling across the edges with labels $D(n)_{i+1}$, $D(n)_{i+2}$, .., $D(n)_{i+h(n+1)}$ where $h(n+1) := 2^{(n+1)-3} = 2^{n-2}$.

The sequence

$$\begin{aligned} &D(n)_{i+1}, D(n)_{i+2}, \ .., D(n)_{i+h(n+1)} = D(n)_{i+1}, \\ &D(n)_{i+2}, \ .., D(n)_{i+2\,h(n)} \end{aligned}$$

will be some rotation of the repeating part r of $D(n)$. Hence we have that sequence $D(n)_{i+1}$, $D(n)_{i+2}$, .., $D(n)_{i+h(n+1)}$ has an odd number of entries in $\{1, 3\}$ (since this sequence is a rotation of r, and r has an odd number of entries in $\{1, 3\}$ by our assumption that n satisfies condition 3).

Recall that $D(n)_{i+1}$, $D(n)_{i+2}$, .., $D(n)_{i+h(n+1)}$ is the sequence of traversed edges within a walk W from $D(n+1)_i$ to $D(n+1)_{i+h(n+1)}$, in the graph shown in Figure B2. Now since this sequence holds an odd number of entries in $\{1, 3\}$, we have that the walk from $D(n+1)_i$ to $D(n+1)_{i+h(n+1)}$ involves crossing an odd number of edges labeled 1 or 3 of the graph shown in Figure B2. However, as seen by examining Figure B2, a directed edge e of the graph has one end in $\{1, 2\}$ and the other end in $\{3, 4\}$ if and only if we have that e is labeled with 1 or 3. In other words, $\forall\, v \in \{1, 2, 3, 4\}$ and $\forall\, e \in \{1, 2, 3, 4\}$ we have that

https://doi.org/10.25088/ComplexSystems.22.1.1

$$\left\lfloor \frac{v}{3} \right\rfloor \neq \left\lfloor \frac{G_e(v)}{3} \right\rfloor \Leftrightarrow e \in \{1, 3\}.$$

It follows that

$$\left\lfloor \frac{D(n+1)_i}{3} \right\rfloor = 1 - \left\lfloor \frac{1}{3} D(n+1)_{i+h(n+1)} \right\rfloor,$$

because $D(n+1)_{i+h(n+1)}$ is obtained by taking $D(n+1)_i$ and doing an odd number of traversals of edges labeled with 1 or 3 (each such edge traversal toggles the value of $\left\lfloor \frac{v}{3} \right\rfloor \in \{0, 1\}$ associated with the current vertex v). Hence we have shown that $D(n+1)$ satisfies condition 2.

Now we will show that $D(n+1)$ satisfies condition 1. To show this, we will use the key fact:

$$\left\lfloor \frac{v}{3} \right\rfloor = \left\lfloor \frac{v'}{3} \right\rfloor \Rightarrow G_e(v) = G_e(v'), \forall\, v, v', e \in \{1, 2, 3, 4\}. \tag{B.5}$$

In other words, the key fact is that if we have a pair of vertices of the graph shown in Figure B2, which are either both in {1, 2} or else both in {3, 4}, then walking along an edge labeled e, starting from either vertex shall lead to the same destination. This can be seen from examining Figure B2.

Now we can use equation (B.5) to prove that $D(n+1)$ satisfies condition 1 as follows.

Since $D(n+1)$ satisfies condition 2, we have $\forall\, j$ that

$$\left\lfloor \frac{D(n+1)_j}{3} \right\rfloor = 1 - \left\lfloor \frac{1}{3} D(n+1)_{j+h(n+1)} \right\rfloor =$$

$$1 - \left(1 - \left\lfloor \frac{1}{3} D(n+1)_{j+2\,h(n+1)} \right\rfloor \right).$$

This implies

$$\left\lfloor \frac{D(n+1)_j}{3} \right\rfloor = \left\lfloor \frac{1}{3} D(n+1)_{j+2\,h(n+1)} \right\rfloor. \tag{B.6}$$

Now, note that

$$D(n+1)_{j+1} = G_{D(n)_{j+1}}\big(D(n+1)_j\big). \tag{B.7}$$

In a similar way, we also have

$$D(n+1)_{j+2\,h(n+1)+1} = G_{D(n)_{j+2\,h(n+1)+1}}\big(D(n+1)_{j+2\,h(n+1)}\big). \tag{B.8}$$

https://doi.org/10.25088/ComplexSystems.22.1.1

Now since $D(n)$ satisfies condition 1 (by assumption), we have that $D(n)$ has period $h(n+1) = 2\,h(n)$ and so we have $D(n)_{j+2\,h(n+1)+1} = D(n)_{j+1}$. We can use this fact to rewrite equation (B.8) as

$$D(n+1)_{j+2\,h(n+1)+1} = G_{D(n)_{j+1}}\big(D(n+1)_{j+2\,h(n+1)}\big). \tag{B.9}$$

Now, since equation (B.6) gives us that

$$\left\lfloor \frac{D(n+1)_j}{3} \right\rfloor = \left\lfloor \frac{1}{3}\, D(n+1)_{j+2\,h(n+1)} \right\rfloor,$$

we may use equation (B.5) to imply that $G_{D(n)_{j+1}}\big(D(n+1)_j\big) = G_{D(n)_{j+1}}\big(D(n+1)_{j+2\,h(n+1)}\big)$. Now we can make substitutions using equations (B.7) and (B.8) to obtain that $D(n+1)_{j+1} = D(n+1)_{j+1+2\,h(n+1)}$, which is our desired result. Since this result holds for any j, we have essentially proved that $D(n+1)$ satisfies condition 1.

Now we will show that $D(n+1)$ satisfies condition 3. For an element $e \in \{1, 2, 3, 4\}$ let us define $S(e) = \{g_e(v) : v \in \{1, 2, 3, 4\}\}$ to be the set of all different destinations in Figure B2 which can be arrived at by starting at some vertex v and then traveling along an edge with label e. For example, $S(1) = \{2, 4\}$ because $G_1(1) = G_1(2) = 4$ and $G_1(3) = G_1(4) = 2$. Now it can easily be checked that we also have $S(2) = \{2, 3\}$, $S(3) = \{2, 4\}$, and $S(4) = \{1, 4\}$.

Now notice that $\forall\, e \in \{1, 2, 3, 4\}$ we have that $S(e)$ has two elements, one of which is in $\{1, 2\}$, and the other of which is in $\{3, 4\}$. The repeating part of $D(n+1)$ has the form:

$$D(n+1)_1\, D(n+1)_2 \ldots D(n+1)_{h(n+1)}\, D(n+1)_{h(n+1)_1} \ldots D(n+1)_{2\,h(n+1)}.$$

Now, for each $i \in \{1, 2, \ldots, h(n+1)\}$ we have that

$$D(n+1)_i = G_{D(n)_i}(D(n+1)_{i-1}), \tag{B.10}$$

by equation (B.3). Also, since $D(n)$ has period $h(n+1)$ we have

$$D(n+1)_{h(n+1)+i} = G_{D(n)_{h(n+1)+i}}\big(D(n+1)_{h(n+1)+i-1}\big) = G_{D(n)_i}\big(D(n+1)_{h(n+1)+i-1}\big). \tag{B.11}$$

Now it follows from equations (B.10) and (B.11) that $D(n+1)_i \in S(D(n)_i)$ and $D(n+1)_{h(n+1)+i} \in S(D(n)_i)$. Moreover, since $D(n+1)$ satisfies condition 2, we have $D(n+1)_i \neq D(n+1)_{h(n+1)+i}$.

https://doi.org/10.25088/ComplexSystems.22.1.1

Since $S(e)$ holds precisely two elements, for each $e \in \{1, 2, 3, 4\}$ it follows that $S(D(n)_i) = \{D(n+1)_i, D(n+1)_{h(n+1)+i}\}$.

We have shown that $D(n)_i$ generates the pair $S(D(n)_i) = \{D(n+1)_i, D(n+1)_{h(n+1)+i}\}$ of values within the repeating part of $D(n+1)$. Note that the sets $S(a) : a \in \{1, 2, 3, 4\}$ have the following properties:

1. If $a \in \{1, 3\}$ then both entries of $S(a)$ are in $\{2, 4\}$.
2. If $a \in \{2, 4\}$ then one of the entries of $S(a)$ is in $\{1, 3\}$ and the other entry of $S(a)$ is in $\{2, 4\}$.

Now let L_n denote the number of entries in $\{1, 3\}$ that occur within the repeating part of $D(n)$, and let M_n denote the number of entries in $\{2, 4\}$ that occur within the repeating part of $D(n)$. Since each entry a in the repeating part of $D(n)$ generates both entries of $S(a)$ within the repeating part of $D(n+1)$, we have that statements 1 and 2 give us the following equations:

$$L_{n+1} = M_n$$
$$M_{n+1} = L_n + M_n$$

Since we are assuming that $D(n)$ satisfies condition 3, we are assuming that L_n and M_n are odd. It hence follows from the given equations that M_{n+1} and L_{n+1} are odd, and so this shows that $D(n+1)$ also satisfies condition 3, as required. And so we have shown that, if $D(n)$ satisfies the lemma then $D(n+1)$ satisfies the lemma. The result can thus be proved by induction with n. □

B.2 Shifting Along Diagonals

Our main objective is to find a substring of a row of the space-time plot of our f-cellular automaton that transforms into our arbitrary binary string β when we apply the substitution rules $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$. Lemma 3 is crucial in this regard because it shows how the properties of the diagonals can be used to move around the space-time plot and find different substrings that yield different outputs under the $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$ operations.

Each substring $Y_i^t\, Y_{i+1}^t \,..\, Y_j^t$ generates a substring

$$\left\lfloor \frac{Y_i^t}{3} \right\rfloor \left\lfloor \frac{Y_{i+1}^t}{3} \right\rfloor .. \left\lfloor \frac{Y_j^t}{3} \right\rfloor$$

when the $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$ substitutions are applied. Lemma 3 allows us to take such a substring $Y_i^t\, Y_{i+1}^t \,..\, Y_j^t$, and con-

https://doi.org/10.25088/ComplexSystems.22.1.1

struct another substring $Y_{i+S[k]}^{t+S[k]}\, Y_{i+1+S[k]}^{t+S[k]} \,..\, Y_{j+S[k]}^{t+S[k]}$. The last $k-1$ characters of this new string are equal to those of the original string. However,

$$\left\lfloor \frac{Y_{k+S[k]}^{t+S[k]}}{3} \right\rfloor = 1 - \left\lfloor \frac{Y_k^t}{3} \right\rfloor.$$

By repeatedly using this construction a substring may be built that gets converted into our arbitrary binary string β by the $\left\lfloor \frac{\cdot}{3} \right\rfloor$ operation. The idea is to keep constructing new strings and increasing the length of the run of rightmost characters that agree with β when the $\left\lfloor \frac{\cdot}{3} \right\rfloor$ operation is applied.

Lemma 3. Let $Y_i^t\, Y_{i+1}^t \,..\, Y_j^t$ be a substring of a row of the space-time plot of the f-cellular automaton, with $t \geq j+2$. Now suppose $k \in \{i, i+1,\ ..,\ j\}$. Consider the substring $Y_{i+S[k]}^{t+S[k]}\, Y_{i+1+S[k]}^{t+S[k]} \,..\, Y_{j+S[k]}^{t+S[k]}$ of a row of the space-time plot, which is obtained by moving $S[k] := 2^{t-k-1}$ places toward the bottom right. Now we have

$$\left\lfloor \frac{Y_{k+S[k]}^{t+S[k]}}{3} \right\rfloor = 1 - \left\lfloor \frac{Y_k^t}{3} \right\rfloor$$

and

$$Y_{m+S[k]}^{t+S[k]} = Y_m^t, \quad \forall\, m \in \{k+1, k+2,\ ..,\ j\}.$$

Proof. Recall that $D(n)_j = Y_j^{n+j-2}$. It follows that the cell Y_j^t lies upon the diagonal $D(n)$ where $n+j-2 = t$, and so $n = 2+t-j$. Since

$$t \geq j+2, \tag{B.12}$$

we have that $n \geq 4$. Now, for each $k \in \{i, i+1,\ ..,\ j\}$ we have that the cell Y_k^t is part of the diagonal $N[k] := 2+t-k$. Since $k \leq j$ we shall have that equation (B.12) implies $N[k] \geq n+j-2 \geq 4$.

According to Lemma 2 we have that the diagonal $D(N[k])$, which Y_k^t is part of, has half-period $h[N[k]] = 2^{N[k]-3} = 2^{t-k-1}$. So now we let $S[k] = h[N[k]] = 2^{t-k-1}$ and Lemma 2 gives us that

$$\left\lfloor \frac{Y_{k+S[k]}^{t+S[k]}}{3} \right\rfloor = \left\lfloor \frac{D(N[k])_{k+h[N[k]]}}{3} \right\rfloor = 1 - \left\lfloor \frac{D(N[k])_k}{3} \right\rfloor = 1 - \left\lfloor \frac{Y_k^t}{3} \right\rfloor.$$

https://doi.org/10.25088/ComplexSystems.22.1.1

Moreover, $\forall\, m \in \{k+1, k+2, \ldots, j\}$ we shall have that Y^t_m lies upon the diagonal $D(N[m])$ where $4 \le N[m] = 2 + t - m < N[k]$. Now it follows (from Lemma 2) that the diagonal $D(N[m])$ has period $2\,h[N[m]] = 2 \times 2^{N[m]-3} = 2^{N[m]-2} = 2^{t-m}$. Since $m > k$ and $S[k] = 2^{t-(k+1)}$, we have that the period $2\,h[N[m]]$ of $D(N[m])$ will be a multiple of $S[k]$. So we have that moving $S[k]$ places to the bottom right, starting from Y^t_m, will correspond to moving along the diagonal that Y^t_m lies upon, a number of places that is a multiple of the period of this diagonal. It follows that this movement will effectively leave the value of the cell unchanged. In other words, we can write $S[k] = q\,2\,h[N[m]]$, for some positive integer q. We then have that $Y^{t+S[k]}_{m+S[k]} = D(N[m])_{m+\mathrm{q2h}[N[m]]} = D(N[m])_m = Y^t_m$. □

B.3 Completing the Proof

We shall describe how to construct a sequence $w(0), w(1), \ldots, w(L)$ of substrings of the space-time plot of our f-cellular automaton. The final string $w(L)$ has the property that

$$\left\lfloor \frac{w(L)_1}{3} \right\rfloor = \beta_1, \left\lfloor \frac{w(L)_2}{3} \right\rfloor = \beta_2, \ldots, \left\lfloor \frac{w(L)_L}{3} \right\rfloor = \beta_L.$$

Once we have constructed this $w(L)$ we can complete our proof because Lemma 7 implies that there is a substring $x(t')_a\, x(t')_{a+1} \ldots x(t')_b$ of system 58 that corresponds to $w(L)$, and this substring is transformed into β by the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$. Before we discuss how to construct this sequence of strings $w(m)$, let us make some definitions.

For a nonempty string $s \in \{1, 2, 3, 4\}^L$, we say that s is *representable* when there exists i, j, t such that $t \ge j + 2 \ge i + 2$ and $s = Y^t_i\, Y^t_{i+1} \ldots Y^t_j$. Essentially, we say that s is representable when it corresponds to a substring of a row of the space-time plot of the f-cellular automaton. The additional condition that $t \ge j + 2 \ge i + 2$ ensures that each part of this substring is contained within a diagonal $D(n)$ for $n \ge 4$.

Also, for $m \in \{0, 1, \ldots, L\}$ we say that such a string $s = s_1\, s_2, \ldots s_L$ is *m-matching with* β when we have that

$$\left\lfloor \frac{s_{L-m+1}}{3} \right\rfloor = \beta_{L-m+1}, \left\lfloor \frac{s_{L-m+2}}{3} \right\rfloor = \beta_{L-m+2}, \ldots, \left\lfloor \frac{s_L}{3} \right\rfloor = \beta_L.$$

https://doi.org/10.25088/ComplexSystems.22.1.1

Essentially we say that s is m-matching with β when the m last characters of s are converted to the last m characters of β by the $\left\lfloor \frac{\cdot}{3} \right\rfloor$ operation.

Our goal is to construct a string $w(L)$ that is representable and L-matching with β. If we can achieve this then we have essentially completed our proof, because $w(L)$ will correspond with a substring of a row of the space-time plot of the f-cellular automata, and $w(L)$ will be converted to β when we apply the $\left\lfloor \frac{\cdot}{3} \right\rfloor$ operation to each of its characters.

Now let us describe how we construct our sequence $w(0), w(1), .., w(L)$ of strings. Our initial string is defined as $w(0) = Y_1^{2+L}\, Y_2^{2+L} .. Y_L^{2+L}$. It is easy to see that $w(0)$ is representable because, letting $i = 1$, $j = L$, and $t = 2 + L$, we have $w(0) = Y_i^t\, Y_{i+1}^t .. Y_j^t$, where $t \geq j + 2 \geq i + 2$.

We can also say (somewhat vacuously) that $w(0)$ is 0-matching with β. What this means is that the final zero characters of $w(0)$ are converted to the final zero characters of β by the $\left\lfloor \frac{\cdot}{3} \right\rfloor$ operation. Any string is 0-matching with β, and so this statement is not really meaningful, but it serves as a base for induction.

Suppose that $m \in \{0, 1, .., L-1\}$ and $w(m)$, are such that $w(m)$ is representable and m-matching with β. Now we will show how to construct a string $w(m+1)$ that is representable and $(m+1)$-matching with β. We construct $w(m+1)$ from $w(m)$ as follows: if

$$\left\lfloor \frac{w(m)_{L-m}}{3} \right\rfloor = \beta_{L-m} \tag{B.13}$$

then we simply let $w(m+1) = w(m)$. In this case $w(m+1)$ clearly is representable and $(m+1)$-matching.

On the other hand, if

$$\left\lfloor \frac{w(m)_{L-m}}{3} \right\rfloor \neq \beta_{L-m},$$

then we must have

$$1 - \left\lfloor \frac{w(m)_{L-m}}{3} \right\rfloor = \beta_{L-m}, \tag{B.14}$$

and in this case our construction of $w(m+1)$ is more elaborate.

In particular, since $w(m)$ is representable, we can write

$$w(m) = w(m)_1\, w(m)_2 .. w(m)_L = Y_i^t\, Y_{i+1}^t .. Y_j^t,$$

https://doi.org/10.25088/ComplexSystems.22.1.1

where $t \geq j+2 \geq i+2$. Note that Y^t_{p-1+i} corresponds to $w(m)_p$ for any p. Now let us define $k = L-m-1+i$. Clearly $Y^t_k = Y^t_{L-m-1+i}$ corresponds to $w(m)_{L-m}$.

Now since $Y^t_i\, Y^t_{i+1} .. Y^t_j$ is such that $t \geq j+2$ and $k \in \{i, i+1, .., j\}$, we may apply Lemma 3 and define $w(m+1)$ as $w(m+1) = Y^{t+S[k]}_{i+S[k]}\, Y^{t+S[k]}_{i+1+S[k]} .. Y^{t+S[k]}_{j+S[k]}$, where $S[k] = 2^{t-k-1}$.

Now we will show that $w(m+1)$ (defined in this way) is $(m+1)$-matching with β. According to Lemma 3, we shall have that

$$\left\lfloor \frac{Y^{t+S[k]}_{k+S[k]}}{3} \right\rfloor = 1 - \left\lfloor \frac{Y^t_k}{3} \right\rfloor \tag{B.15}$$

and

$$Y^t_{k+1}\, Y^t_{k+2} .. Y^t_L = Y^{t+S[k]}_{k+1+S[k]}\, Y^{t+S[k]}_{k+2+S[k]} .. Y^{t+S[k]}_{L+S[k]}. \tag{B.16}$$

Since Y^t_k corresponds to $w(m)_{L-m}$ and $Y^{t+S[k]}_{k+S[k]}$ corresponds to $w(m+1)_{L-m}$, we have that equations (B.14) and (B.15) imply

$$\begin{aligned} &\left\lfloor \frac{w(m+1)_{L-m}}{3} \right\rfloor = \left\lfloor \frac{Y^{t+S[k]}_{k+S[k]}}{3} \right\rfloor = \\ &1 - \left\lfloor \frac{Y^t_k}{3} \right\rfloor = 1 - \left\lfloor \frac{w(m)_{L-m}}{3} \right\rfloor = \beta_{L-m}. \end{aligned} \tag{B.17}$$

Moreover, since $w(m)$ is m-matching with β we have

$$\begin{aligned} \beta_{L-m+1}\, \beta_{L-m+2} .. \beta_L &= \left\lfloor \frac{w(m)_{L-m+1}}{3} \right\rfloor \left\lfloor \frac{w(m)_{L-m+2}}{3} \right\rfloor .. \\ \left\lfloor \frac{w(m)_L}{3} \right\rfloor &= \left\lfloor \frac{Y^t_{k+1}}{3} \right\rfloor \left\lfloor \frac{Y^t_{k+2}}{3} \right\rfloor .. \\ \left\lfloor \frac{Y^t_L}{3} \right\rfloor = \left\lfloor \frac{Y^{t+S[k]}_{k+1+S[k]}}{3} \right\rfloor &\left\lfloor \frac{Y^{t+S[k]}_{k+2+S[k]}}{3} \right\rfloor .. \left\lfloor \frac{Y^{t+S[k]}_{L+S[k]}}{3} \right\rfloor = \\ &\left\lfloor \frac{w(m+1)_{L-m+1}}{3} \right\rfloor \left\lfloor \frac{w(m+1)_{L-m+2}}{3} \right\rfloor .. \\ &\left\lfloor \frac{w(m+1)_L}{3} \right\rfloor, \end{aligned} \tag{B.18}$$

https://doi.org/10.25088/ComplexSystems.22.1.1

where we use equation (B.16) to go from the third line to the fourth line. Together equations (B.17) and (B.18) imply that $w(m+1)$ is $(m+1)$-matching with β.

To see that $w(m+1) = Y_{i+S[k]}^{t+S[k]}\, Y_{i+1+S[k]}^{t+S[k]} \,..\, Y_{j+S[k]}^{t+S[k]}$ is representable, note that we can define $t' = t + S[k]$, $i' = i + S[k]$, and $j' = j + S[k]$. Now clearly $w(m+1) = Y_{i'}^{t'}\, Y_{i'+1}^{t'} \,..\, Y_{j'}^{t'}$, where $t' \geq j' + 2 \geq i' + 2$ (because $t \geq j + 2 \geq i + 2$). This shows that $w(m+1)$ is representable.

So now we have shown that $w(0)$ is representable and 0-matching with β. Also, we have shown that $\forall\, m \in \{0, 1, \,.., L-1\}$ we have that if $w(m)$ is representable and m-matching with β then we can construct a string $w(m+1)$ that is representable and $(m+1)$-matching with β. It therefore follows, by induction with m, that we can construct a string $w(L) \in \{1, 2, 3, 4\}^L$ that is representable and L-matching with β.

Since $w(L)$ is representable, it corresponds to a substring of a row of the space-time plot of our f-cellular automaton, in the sense that there exist i^*, j^*, and t^* such that $w(L) = Y_{i^*}^{t^*}\, Y_{i^*+1}^{t^*} \,..\, Y_{j^*}^{t^*}$. According to Lemma 1, there is a substring $x(t^*+3)_a\, x(t^*+3)_{a+1} \,..\, x(t^*+3)_b$ of $x(t^*+3)$ that is converted into $w(L)$ by applying the replacement rules $10 \to 1$, $110 \to 2$, $101 \to 3$, $01 \to 4$. Moreover, since $w(L)$ is L-matching with β we have that $w(L)$ gets converted into β by the substitution operations $1 \to 0$, $2 \to 0$, $3 \to 1$, $4 \to 1$. It follows that the substring $x(t^*+3)_a\, x(t^*+3)_{a+1} \,..\, x(t^*+3)_b$ is transformed directly into β by the replacement rules $10 \to 0$, $110 \to 0$, $101 \to 1$, $01 \to 1$, which completes the proof. □

Acknowledgments

We gratefully acknowledge support from EPSRC Grant EP/D003105.

References

[1] N. Dershowitz and J. Jouannaud, "Rewrite Systems," in *Handbook of Theoretical Computer Science*, Vol. B: Formal Methods and Semantics (J. Van Leeuwen, ed.), Universitè Paris-Sud, Centre d'Orsay, Laboratoire de recherche en Informatique, 1989.

[2] M. Morse and G. Hedlund, "Unending Chess, Symbolic Dynamics and a Problem in Semigroups," *Duke Mathematical Journal*, **11**(1), 1944 pp. 1–7. doi:10.1215/S0012-7094-44-01101-4.

https://doi.org/10.25088/ComplexSystems.22.1.1

[3] A. Lindenmayer, "Mathematical Models for Cellular Interaction in Development," *Journal of Theoretical Biology*, **18**, 1968 pp. 280–315.

[4] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, New York: Springer-Verlag, 1991.

[5] P. Worth and S. Stepney, "Growing Music: Musical Interpretations of L-Systems," in *Applications of Evolutionary Computing*, Vol. 3449 (F. Rothlauf et al., eds.), Berlin: Springer-Verlag, 2005 pp. 545–550. doi:0.1007/978-3-540-32003-6_56.

[6] S. Goel and I. Rozehnal, "Some Non-biological Applications of L-Systems," *International Journal of General Systems*, **18**(4), 1991 pp. 321–405. doi:10.1080/03081079108935155.

[7] G. S. Hornby and J. B. Pollack, "Evolving L-Systems to Generate Virtual Creatures," *Computers & Graphics*, **25**(6), 2001 pp. 1041–1048. doi:10.1016/S0097-8493(01)00157-1.

[8] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[9] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D: Nonlinear Phenomena*, **10**(1–2), 1984 pp. 1–35. doi:10.1016/0167-2789(84)90245-8.

[10] A. Smith. "Universality of Wolfram's 2, 3 Turing Machine." Submitted for the Wolfram 2, 3 Turing Machine Research Prize, 2007. http://www.wolframscience.com/prizes/tm23/TM23Proof.pdf.

[11] C. Bennet and R. Landauer, "The Fundamental Physical Limits of Computation," *Scientific American*, **253**, 1985 pp. 48–56. http://www.scientificamerican.com/article.cfm?id=the-fundamental-physical-limits-of-computation.

[12] K. Jarkko, "Reversible Cellular Automata," in *Developments in Language Theory: Proceedings of the 9th International Conference (DLT05)*, Palermo, Italy (C. De Felice and A. Restivo, eds.), Berlin: Springer, 2005 pp. 57–68.

[13] M. Seredynski and B. Pascal, "Block Cipher Based on Reversible Cellular Automata," *New Generation Computing*, **23**(3), 2005 pp. 245–258. doi:10.1007/BF03037658.

[14] M. Nelson, *Data Compression Book*, Redwood City, CA: M&T Books, 1991.

[15] C. Chaitin, "Algorithmic Information Theory," *IBM Journal of Research and Development*, **21**, 1977 pp. 350–359. http://www.cs.auckland.ac.nz/~chaitin/ibm.pdf.

[16] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, **24**(5), 1978 pp. 530–536. doi:10.1109/TIT.1978.1055934.

https://doi.org/10.25088/ComplexSystems.22.1.1

[17] R. Cilibrasi and P. M. B. Vitanyi, "Clustering by Compression," *IEEE Transactions on Information Theory*, **51**(4), 2005 pp. 1523–1545. doi:10.1109/TIT.2005.844059.

[18] H. Zenil, "Compression-Based Investigation of the Dynamical Properties of Cellular Automata and Other Systems," *Complex Systems*, **19**(1), 2010 pp. 1–28. http://www.complex-systems.com/pdf/19-1-1.pdf.

[19] G. Chaitin, *Meta Math! The Quest for Omega*, New York: Pantheon Books, 2005.

https://doi.org/10.25088/ComplexSystems.22.1.1