

Reflexive Composition of Elementary State Machines, with an Application to the Reversal of Cellular Automata Rule 90

Chris Salzberg^{1,*} and Hiroki Sayama²

¹Shopify

²Binghamton University, State University of New York

Abstract

We explore the dynamics of a one-dimensional lattice of state machines on two states and two symbols sequentially updated via a process of “reflexive composition.” The space of 256 machines exhibits a variety of behavior, including substitution, reversible “billiard ball” dynamics, and fractal nesting. We show that one machine generates the Sierpiński Triangle and, for a subset of boundary conditions, is isomorphic to cellular automata Rule 90 in Wolfram’s naming scheme. More surprisingly, two other machines follow trajectories that map to Rule 90 in reverse. Whereas previous techniques have been developed to uncover preimages of Rule 90, this is the first study to produce such inverse dynamics naturally from the formalism itself. We argue that the system’s symmetric treatment of *state* and *message* underlies its expressive power.

Introduction

The question of how open-ended complexity can emerge from simple rules is foundational to the fields of both Complex Systems and Artificial Life. Models employed to reproduce such emergence commonly take the form of a discrete set of rules applied to a finite lattice of states. Cellular Automata (CA), the most popular of these models, have been shown to exhibit a surprisingly rich variety of evolutionary and lifelike behavior (Sayama and Nehaniv, 2025). Elementary Cellular Automata (ECA), two-state CA that evolve on a one-dimensional lattice, are arguably the most basic and minimal example of the emergence of complexity from simple, discrete rules (Wolfram, 2002).

Implicit in these popular models is a divide between, on the one hand, the *states* of a system and, on the other hand, the *rules* that apply to those states. The earliest abstract model of computation, the Turing Machine, embodies this distinction in its division of the world into *machine* and *tape* (Turing, 1937); Church’s λ -calculus contains a similar divide between the *function* and its *argument* (Church, 1936). There is a conundrum, however, in the fact that we find no physical equivalent in nature to the rules and states of these artificial systems. Despite intriguing parallels, there

are no cleanly distinguishable “machine molecules” or “tape molecules” in the complex mechanisms of DNA/RNA transcription, for example. Indeed, the assumption that such distinct categories *must exist* limits the capacity of these systems to model their emergence from within the given frame of reference.

The notion of *emergence of function*, unrepresented at the core of these rule-based systems, is nevertheless of central importance in drawing analogies between computational models and the physical world. The lack of a model to express such emergence inspired the “algorithmic chemistry” model of Fontana and Buss, which employs LISP expressions that interact algorithmically in a fixed ensemble (Fontana, 1991; Fontana and Buss, 1994). The notion has a powerful interpretation in molecular biology, where it serves as an alternative to dogmatic theories of information “carriers” and “processors” (Wills, 1989). Origin-of-life problems, in which the distinction between carriers and processors is not yet well defined, are an area where the study of such emergence in a computational model could provide valuable insight. The problem space, however, has eluded widespread attention and remains largely underdeveloped.

In this paper we explore the dynamics of a discrete system defined by a one-dimensional lattice of states and a finite-state machine applied to update them. Unlike cellular automata, the update process, referred to as “reflexive composition,” is symmetric in its treatment of *state* and *message*. This symmetry results in the surprising finding that certain elementary cellular automata, in particular Rule 90 in Wolfram’s naming scheme, are represented in the system alongside their inverse (time reversed process). This inverse process is not expressible as cellular automata but arises naturally from the formulation presented. Furthermore, we show that for this rule, *time reversal* is equivalent to the inversion of *states* and *messages* in the system’s FSM. Interpretations of this finding are presented in the Conclusions section.

Formulation

The formulation of the “composite state machine” was introduced in earlier studies on a graph-based artificial chemistry

* Corresponding author: me@chrisalzberg.com

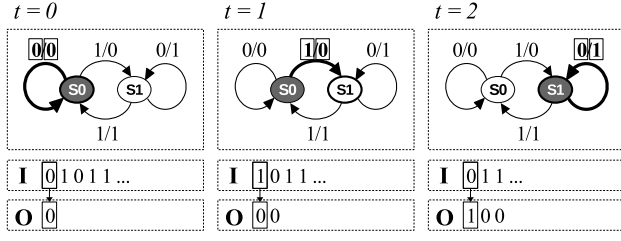


Figure 1: Three steps of FSM with $Q = \{S0, S1\}$ and $\mathcal{A} = \{0, 1\}$. State-transition function δ and output function φ are defined in Figure 4(a). Dark nodes and bold edges indicate the current state and transition, respectively, in each frame, with input and output streams shown below.

(Salzberg, 2006a,b, 2007). The focus in those studies was on state-space complexity; here our focus will be on exploring *trajectories* in the state space, an aspect overlooked in previous work. In this section, we review the formulation, introduce a naming scheme for minimal state machines, and present a new visualization of trajectories that makes it easier to see the similarities with a subset of ECA.

The finite state machine

The basis for the formulation is the traditional finite state machine (FSM) from computer science theory (Minsky, 1967). A FSM is described in terms of a state-transition graph $G = (Q, \mathcal{A}, \delta, \varphi)$, where Q is a set of states, \mathcal{A} a finite alphabet, $\delta : \mathcal{A} \times Q \rightarrow Q$ a transition function, and $\varphi : \mathcal{A} \times Q \rightarrow \mathcal{A}$ an output function. We refer to the transition and output functions for an input message $i \in \mathcal{A}$ from a state $q \in Q$ in the forms $\delta_q(i) \equiv \delta(i, q)$ and $\varphi_q(i) \equiv \varphi(i, q)$, respectively. The FSM constitutes a minimal abstraction of a physical machine: a collection of states, transitions between states, and elementary input/output mappings. Given an input $i \in \mathcal{A}$, a FSM in state $q \in Q$ follows the transition to state $\delta_q(i)$ and returns an output $\varphi_q(i)$.

Here we consider the evolution of an FSM over time, with $i(t) \in \mathcal{A}$ representing an input stream, $q(t) \in Q$ a dynamic state, and $o(t) \in \mathcal{A}$ an output stream. The next state is then defined as $q(t+1) := \delta_q(i(t))$, and the output as $o(t) := \varphi_q(i(t))$. The evolution of a FSM is shown in Figure 1 for the first three inputs (0, 1, 0) of an input sequence.

Naming scheme for elementary state machines

To simplify the presentation of results, we introduce a naming scheme for the elementary state machines to be explored in later sections. Whereas previous work considered more complex state machines, our focus here will be on the minimal case of two states $Q = (S0, S1)$ and two symbols $\mathcal{A} = (0, 1)$. We can enumerate such machines by considering that for any particular machine, we must define state transition functions (δ_{S0} and δ_{S1}) and output functions (φ_{S0}

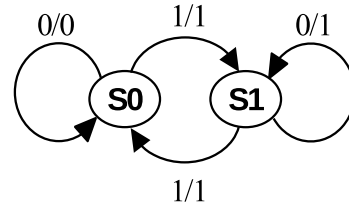
and φ_{S1}) for two states, each taking and producing one of two possible values ($(0, 1)$ or $(S0, S1)$). Thus each function requires two bits to represent, and we have four such functions to define, for a total of 8 bits of information and 256 possible machines. This is shown in Table 1, with an example FSM, *M61*, shown in Figure 2 along with its encoding.

Although there are a total of 256 possible elementary state machines, many are equivalent computationally to others in the same set. The choice of the message values 0 and 1 is arbitrary, so each machine has a “complementary machine” with these values reversed. Likewise, the states of a machine can be reversed without changing the dynamics of the system, yielding an equivalent “mirror machine”. Finally, both message values and states can be swapped to yield a “complementary mirror machine.” Thus any FSM is computationally equivalent to as many as three other machines.

Of the set of all 256 elementary FSMs, only 76 are unique under these transformations. This is similar to the equivalence among ECA rules, of which 88 are computationally unique. In Table 2, we list the interesting FSMs of focus in this study and, where applicable, their equivalent CA rules.

State	S0				S1			
Input	0		1		0		1	
Function	δ	φ	δ	φ	δ	φ	δ	φ
Encoding Bit	7	6	5	4	3	2	1	0

Table 1: Naming scheme for two-state, two-input FSM. The results of the functions δ and φ for each pair of arguments are concatenated into a binary string and converted to decimal, similar to the naming of ECA rules.



State	S0				S1			
Input	0		1		0		1	
Destination / Output	S0	0	S1	1	S1	1	S0	1
Encoding	0	0	1	1	1	1	0	1

Figure 2: FSM with identifier *M61*. States *S0* and *S1* and message values 0 and 1 are each mapped to the bits 0 and 1, respectively, resulting in the bit string $(00111101)_2 = 61$.

Reflexive composition

With the FSM and its naming scheme defined, we now consider a process by which to evolve a set of states on a lat-

tice. Consider first a pair of states $q_s(t)$ (sender) and $q_r(t)$ (receiver) in Q ; this will later be generalized to a lattice of arbitrary many states. In contrast to previous work, which considered the more general case of distinct sender and receiver state machines, here we assume the lattice is uniform, which will simplify our presentation. Sender and receiver states are thus both updated using the same FSM.

At each step, we calculate the output for the pair of states by composing the output functions at q_s and q_r , φ_s and φ_r , to produce the lattice (row) output $o(t)$ for an input $i(t)$:

$$o(t) = \varphi_r(\varphi_s(i(t))) \quad (1)$$

Thus the output of the state pair is calculated simply by “piping” the output of the sender to the input of the receiver.

The state transition update employs a similar composition, incorporating the transition functions δ_s and δ_r at q_s and q_r , and the output function φ_s , to produce:

$$\begin{aligned} q_r(t+1) &= \delta_s(i(t)), \\ q_s(t+1) &= \delta_r(\varphi_s(i(t))). \end{aligned} \quad (2)$$

These equations advance each state in the lattice (here, q_s and q_r) along transitions corresponding to the input $i(t)$ and its composition $\varphi_s(i(t))$, passed as arguments to δ_s and δ_r , respectively.

On the left side of eq. (2), we assign the results of these transitions to the next step states $q_s(t+1)$ and $q_r(t+1)$, but in doing so alternate the identity of sender and receiver by passing the result of δ_s to q_r and δ_r to q_s . The alternation of sender and receiver in the update process, detailed extensively in Salzberg (2006b, 2007), is essential to ensuring that no element of the system acts exclusively as information processor or information carrier. We refer to this symmetry between sender and receiver as “reflexivity”.

Four steps in the evolution of a sender/receiver pair of states is shown in Figure 3 for the FSM with identifier *M45*. We adopt the convention of maintaining the position of persisted states by horizontally flipping sender and receiver at each step; this ensures that state changes visually align along columns of the grid and makes transitions easier to follow. It also highlights the spatial symmetry which flipping enforces on the lattice, a mechanism we refer to as “folding.”

We can now generalize the reflexive composition described above to an arbitrarily-sized lattice. In earlier work, this generalization took the form of repeated application of composition to generate exponentially larger state-space graphs; here our focus will be on a fixed lattice of n machines, but the update process is identical.

For a lattice of n machines, the output function is applied across the full set of output functions:

Thus in this figure, the first and third row have the sender on the left and receiver on the right, while the second and fourth row have the receiver on the left and sender on the right.

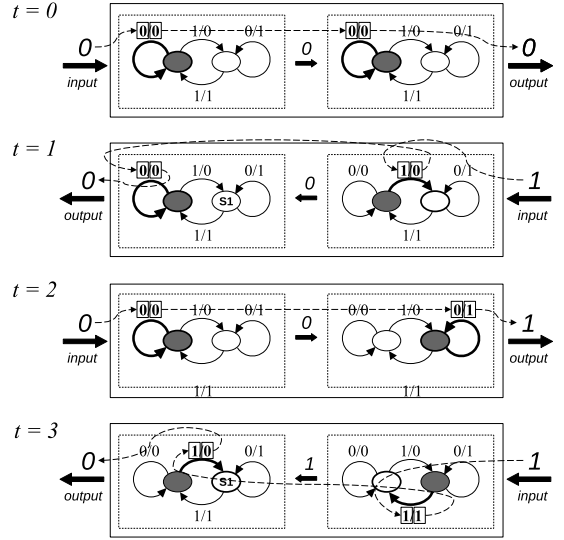


Figure 3: Four steps (vertical) of state composition (horizontal) applied to a two cell lattice of *M45* for input sequence $(0, 1, 0, 1)$. Directionality is alternated at each step. Dark nodes and bold edges indicate the current state and transition, respectively. Dashed lines show the composition of inputs to outputs at each step; row outputs are discarded.

$$o(t) = (\varphi_n \circ \varphi_{n-1} \circ \dots \circ \varphi_0)(i(t)) \quad (3)$$

The transition function generalizes in the same way, by application of state composition and reversal across the full lattice of n states:

$$\begin{aligned} q_n(t+1) &= \delta_0(i(t)) \\ q_{n-1}(t+1) &= \delta_1(\varphi_0(i(t))) \\ &\dots \\ q_0(t+1) &= (\delta_n \circ \varphi_{n-1} \circ \dots \circ \varphi_0)(i(t)) \end{aligned} \quad (4)$$

Equations (3) and (4) reduce to (1) and (2) when $n = 2$. Three steps of reflexive composition applied to a 5-state lattice of *M45* machines is shown in Figure 4.

The reader is encouraged to familiarize themselves with the update process shown here; in the next section we omit the details of state transitions at each cell to focus on the dynamics of state changes across much larger lattice sizes.

Results

Consider now the full space of elementary FSMs updated using the reflexive composition process described in the last section. Figure 5 tabulates trajectories for each of these 256 machines starting from a 19-cell lattice with a single

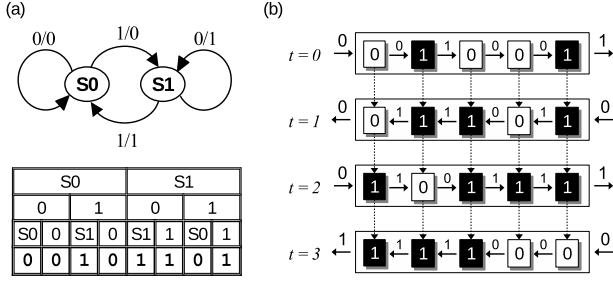


Figure 4: (a) Top: Two-state, two-symbol state machine identified as *M45*. Bottom: table of state machine transitions and derivation of naming $((00101101)_2 = 45)$. Bottom: table of state machine transitions. (b) Four steps of the state composition trajectory of a five-cell *M45* lattice with null-input boundary conditions and initial states $(0, 1, 0, 0, 1)$.

FSM	Equivalents	Notes
7	47, 88, 218	ECA: R128, R254.
44	104, 199, 214	Billiard ball dynamics. Figure 8.
45	120, 135, 210	ECA: R90, R165. Figs. 3, 4. & 6.
54	99, 156, 201	Reverse R90. Figs. 10, 12, 13 & 15.
60	105, 150, 195	Reverse R90. Figs. 10, 11 & 13.
61	121, 131, 146	Substitution system. Figs. 2 & 9.

Table 2: Selection of elementary FSMs and their equivalent mirrors and complements. There are 76 unique machines in total.

centered *S1*-state cell surrounded by *S0*-state cells, with “0-input” boundary conditions ($i(t) = 0$ for all t).

Although many of the trajectories in Figure 5 are blank (all *S0* or *S1* states), many others exhibit diverse and complex behavior, summarized in Table 2. To help explore this behavior, we begin by partitioning the space of machines in such a way as to clarify its relation to cellular automata.

State reporting and message propagation

We refer to a FSM as *state-reporting* if it satisfies the requirement that it always “reports its state”, i.e.:

$$\forall i \in \mathcal{A} : \varphi_q(i) = O_q \quad (5)$$

for some constant output $O_q \in \mathcal{A}$ that is independent of the input i but unique for different states $q \in \mathcal{Q}$.

An example of a state-reporting FSM is *M45*, shown in Figure 4: regardless of its input, for this machine φ_{S0} returns 0 and φ_{S1} returns 1. *M45* thus always conveys its previous state (before transition) to the next cell in the lattice. It is not hard to see that the property of a FSM being state-reporting

In contrast to its output $o(t)$, the state $q(t)$ of such a machine at step t can be altered by its input at that step. Propagation of messages across the lattice must thus occur through state change in this type of machine.

limits its action at a distance to only directly neighboring cells in the lattice. Consequently, this property is a requirement for equivalence between FSMs under reflexive composition and synchronous cellular automata rules. We show an example of this equivalence later in this section.

The opposite of a state-reporting machine is one which, in at least one case, reports the content of its input in its output. We refer to such machines as *message-propagating* and define them as the negation of (5):

$$\exists q \in \mathcal{Q}, i_1, i_2 \in \mathcal{A} : \varphi_q(i_1) \neq \varphi_q(i_2)$$

In other words, a *message-propagating* FSM is one for which a different input can result in a different output for the same state q . FSMs like this can exert action at a distance and are in general not relatable to synchronous cellular automata. *M61* from Figure 2 is an example of a message-propagating FSM since $\varphi_{S0}(0) \neq \varphi_{S0}(1)$ for this machine.

Equivalence to cellular automata and M45

It is easy to show that two steps of a state-reporting FSM under reflexive composition are equivalent to one step of a synchronous ECA. In Figure 6, we show that *M45*, seen earlier in Figure 4, is equivalent to CA Rule 90. Intuitively, this equivalence makes sense given that *M45* is the minimal representation of an adding machine modulo 2, and reflexive composition applies this addition twice, once in each direction; this corresponds to the XOR operation of CA Rule 90.

Boundary conditions in this equivalence relation require special attention, however. We can ensure that the boundaries on alternating steps act like a constant 0-valued CA cell by extending the lattice by two “virtual states”, which act like *S0* states on those steps (but not necessarily on other steps). In Figure 7, we illustrate this technique of enforcing CA boundaries on even steps.

The technique requires a different calculation for even and odd steps. Since the left-hand virtual boundary is *S0*, the input to the lattice at step t is 0 (since *M45* reports its state). The right-hand virtual boundary cell is updated according to *M45*’s transition function applied to the lattice output $o(t)$: if $o(t)$ is 0, it remains *S0*, if it is 1, it transitions to *S1*. Since *M45* reports its state, the virtual boundary cell thus sends 0 as the right-hand input to the next step $i(t+1)$ if it remained in state *S0*, otherwise it sends 1.

This can be condensed to simply:

$$\begin{aligned} i(t_{\text{even}}) &= 0 \\ i(t_{\text{odd}}) &= o(t-1) \end{aligned}$$

This is equivalent to taking the sum modulo 2 of the previous row (with *S0* and *S1* mapped to 0 and 1, respectively). Applying these boundary conditions to a lattice of *M45* machines with reflexive composition results in a trajectory identical to ECA Rule 90 for even steps. For odd steps,

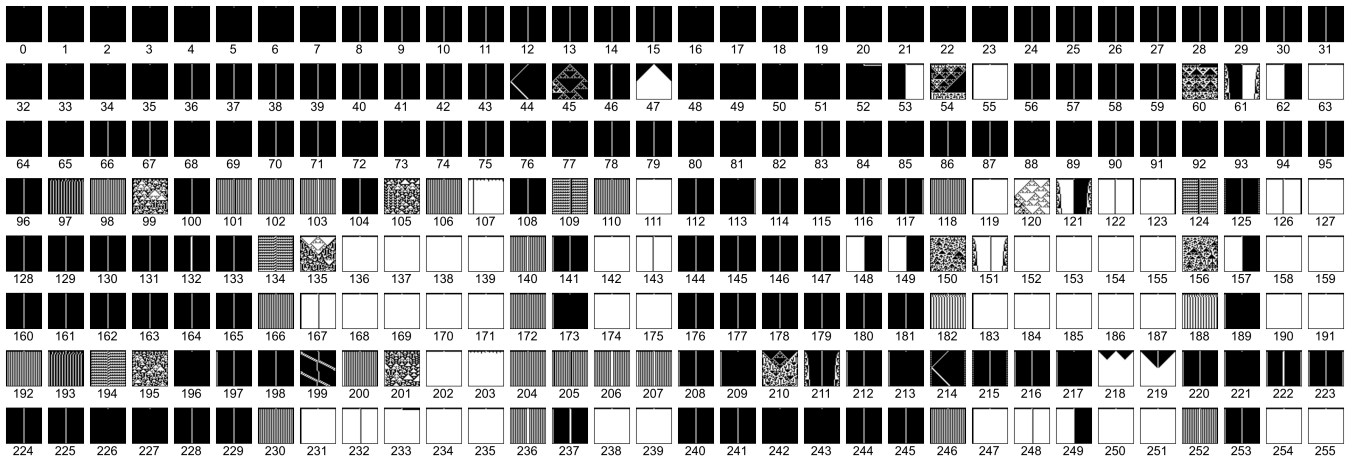


Figure 5: Trajectories of the full collection of 256 2-state, 2-symbol machines from initial centered pixel on 19-cell lattice with input-0 boundary conditions. For visibility only even rows are shown. Black = $S0$, White = $S1$.

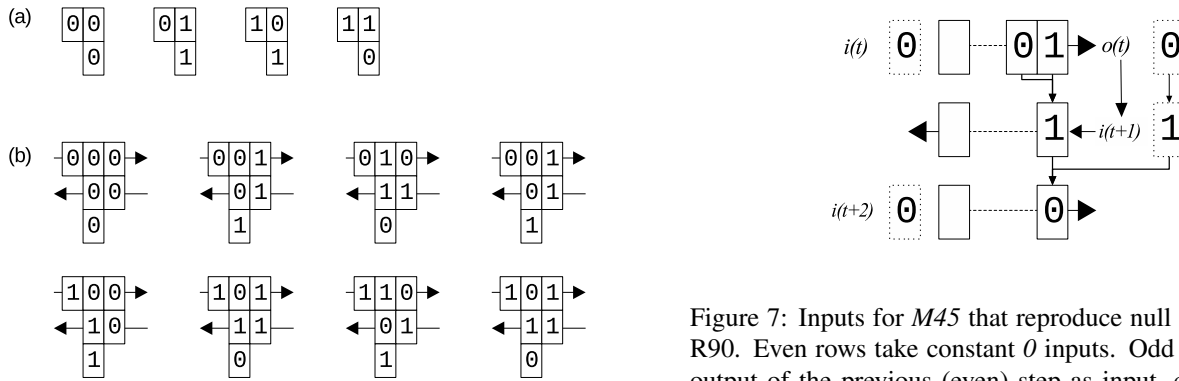


Figure 6: Equivalence between alternate steps of $M45$ and each step of CA Rule 90. (a) A simplified view of $M45$ purely as a 1-neighbor CA with directionality flipped at each step. (b) Mapping from $M45$ transitions to CA Rule 90.

it will diverge from R90 in cases where there are 1 cells on the boundary of even steps.

Message-propagating FSMs

We have focused so far on state-reporting FSMs, which as shown for $M45$, are equivalent on alternating steps to ECA rules. Message-propagating FSMs, in contrast, can act at a distance across the lattice and are thus not reproducible by synchronous cellular automata. In this section we investigate two interesting cases in this class of FSMs.

M44: Billiard ball dynamics Figure 8 plots the trajectory of $M44$ over four hundred steps from an initial random lattice of cells. The dynamics of this rule are reminiscent of invertible cellular automata (Toffoli and Margolus, 1990), which commonly have billiard-ball like dynamics. Indeed,

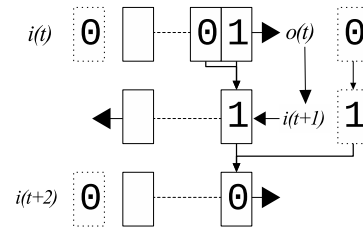


Figure 7: Inputs for $M45$ that reproduce null boundaries of R90. Even rows take constant 0 inputs. Odd rows take the output of the previous (even) step as input, corresponding to the value that the “virtual boundary state” (dotted boxes) was updated to from its even-row value of 0.

$M44$ is reversible and can be inverted simply by flipping the lattice of states at any step in the trajectory.

M61: Substitution system For given initial conditions and boundary inputs, $M61$, shown earlier in Figure 2, produces the patterns of a substitution system along its boundaries (Wolfram, 2002).

M54 and M60: Reversal of CA Rule 90

Reflexive composition produces its most intriguing result in the dynamics of message-propagating machines $M54$ and $M60$, shown in Figure 10. Figure 11 charts trajectories for (identical) random initial conditions and null-input boundary conditions. The striking complexity of these trajectories stands out among FSMs explored in this study. While reminiscent of ECA, the simultaneous change of state across disparate sites of the lattice is impossible in synchronous CA.

Closer inspection of the state-change dynamics of $M54$ reveals that this machine is in fact the reverse of $M45$ de-

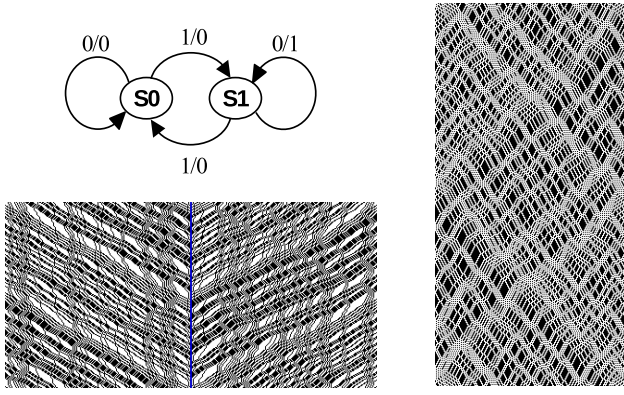


Figure 8: Top-left: $M44$ state transition diagram. Right: Four hundred steps of $M44$ with initial random 200-state configuration and null input boundaries. Bottom-left: same steps, split into even rows (left) and odd rows (right).

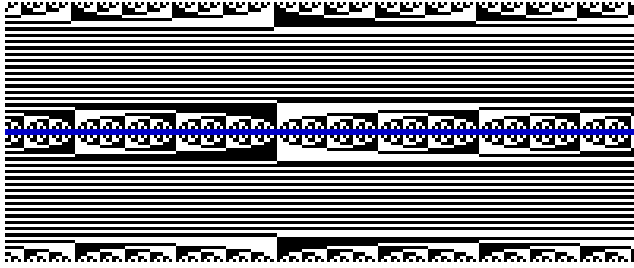


Figure 9: 400 steps of $M61$ with initial configuration of alternating 0s and 1s and null-input boundaries, split into even (top) and odd (bottom) rows. Time runs from left to right.

scribed earlier, and as such that it reproduces the reverse of CA Rule 90 (see Figure 12.) It is known that R90 is not reversible on a periodic finite lattice if the number of sites in the lattice with value 1 is odd (Martin et al., 1984, p. 227). While here we consider null boundary conditions, the same constraint applies to cases where periodicity does not affect results. To avoid irreversible configurations, we thus consider first a symmetric arrangement of initial states in which non-quiescent patterns do not reach the (null) boundaries. As shown in Figure 12 for an initial condition of two centered 1 cells, under these conditions $M54$ indeed produces the reverse-time trajectory of Rule 90. (Later steps require special treatment of boundaries, see below.)

In Figure 13 we compare the dynamics of $M54$ and $M60$ with the same initial states and boundary conditions. The lattice configurations of these two machines are identical on even steps and shifted by one cell on odd steps. (We focus below on $M54$, but $M60$ produces equivalent dynamics.)

Reproducing reverse dynamics of R90 for the more general case requires correct treatment of boundary conditions. This is shown in Figure 14, again using “virtual” states to represent null boundaries. We use the fact that $M54$ trans-

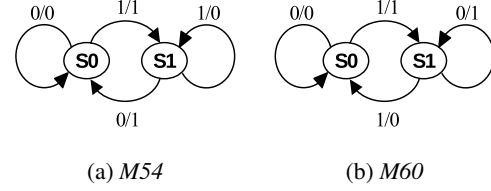


Figure 10: Transition diagrams of $M54$ and $M60$. FSMs differ only in the inversion of their destinations from $S1$.

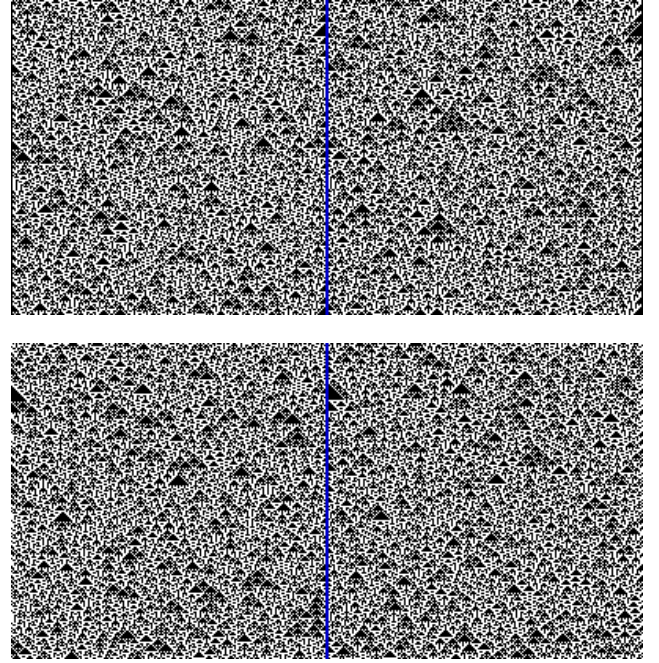


Figure 11: 400 steps of $M54$ (top) and $M60$ (bottom) with the same random initial conditions, split into even (left) and odd (right) steps.

lates its input into the destination state (0 transitions the state to $S0$, 1 transitions it to $S1$). Thus in order to satisfy the condition that even rows must have a fixed $S0$ state starting the row, the output of the previous row $o(t-1)$ must equal 0 .

Similarly, again using the fact that $S0$ translates its input into its destination state, a 0 output for $o(t)$ will transition the right-hand boundary state to $S0$, and a 1 output to $S1$. The requirement that this transition back to $S0$ in the next step constrains $i(t+1)$ to be 0 if $o(t)$ was 0 and 1 if $o(t)$ was 1 .

We can summarize these boundary conditions as:

$$i(t_{\text{odd}}) = o(t-1) \quad (6)$$

$$o(t_{\text{odd}}) = 0 \quad (7)$$

These constraints are applied using an algorithm in which $i(t) = 0$ is first attempted, and if unsuccessful $i(t) = 1$ is

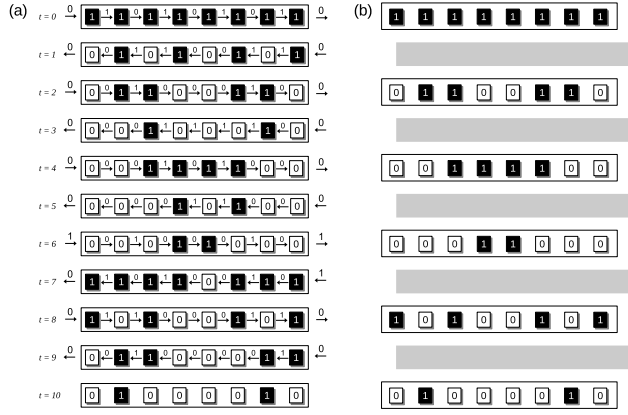


Figure 12: Evolution of *M54* applied to an 8-cell lattice. (a) Even and odd steps. (b) Even steps only, highlighting isomorphism to R90 in reverse. Up to $t = 5$ inputs are all 0, but the null constraints of eqs. (6) and (7) demand that the input of $t = 6$ and $t = 7$ are both 1 to ensure that $o(6) = i(7)$ and $o(7) = 0$.

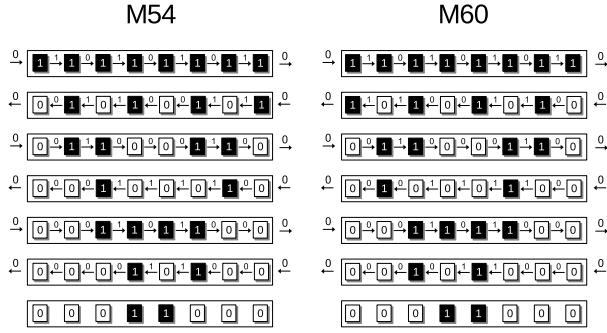


Figure 13: Comparison of *M54* and *M60* for the same initial configuration as Figure 12. Even steps are identical; odd steps of *M60* are one cell to the left of *M54*.

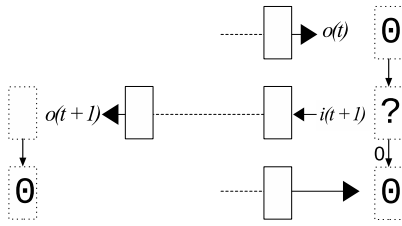


Figure 14: Inputs for *M54* that reproduce null boundaries of R90 in reverse. Output of the even step t , $o(t)$, transitions the right-hand "virtual boundary cell" such that $i(t+1)$ must be the same value to transition it back to $S0$ at the next even step $t+2$. The odd step output $o(t+1)$ must be zero to transition the left-hand state to $S0$.

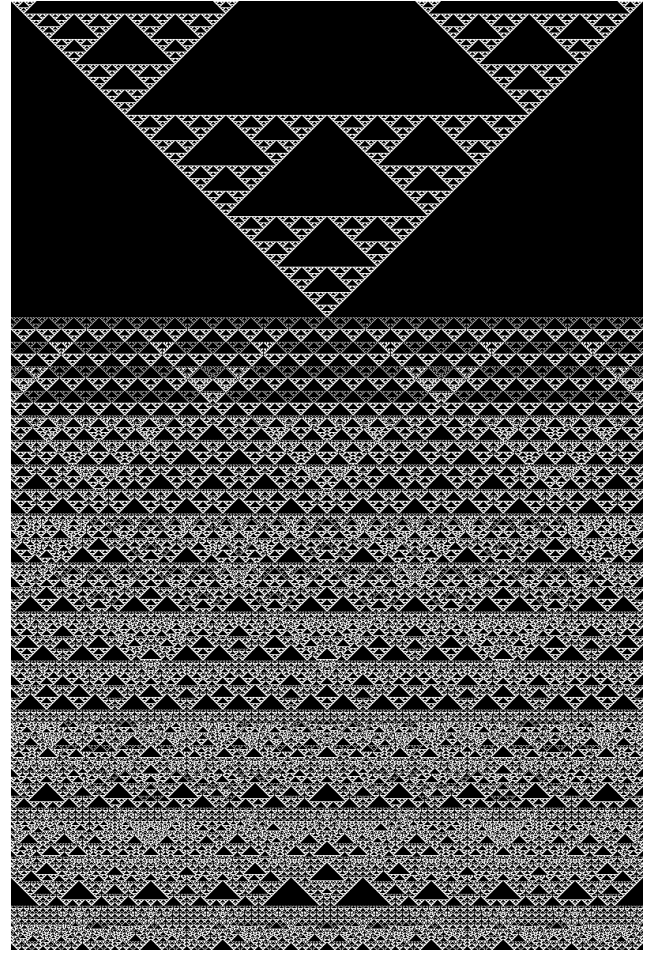


Figure 15: Even steps of 2400-step *M54* trajectory on 800-state lattice initiated with an intermediate R90 configuration. Inputs reproduce forward-direction R90 null boundaries.

used instead. This technique is used to generate boundary inputs in Figure 12. A larger grid applying these boundary constraints produces the trajectory in Figure 15.

While the initial portion of this trajectory is well known, the transition to preimages of the original Rule 90 configuration reveals a striking wealth of complexity not reported, to our knowledge, in any previous studies. The sudden transitioning of entire rows of null cells at once, also exhibited at a smaller scale in Figure 12, is made possible by the fact that *M54* is message-propagating. Unlike the forward direction state-reporting machine *M45* (and Rule 90), this reverse direction can propagate messages across multiple cells in a single step; this accounts for coordinated state changes across arbitrary distances in the lattice. The specific arrangement of cells in steps following this transition is moreover strongly affected by lattice size, with even small changes producing entirely different pattern dynamics. Figure 16 highlights this effect on two lattices with different sizes but identical initial configurations, each producing

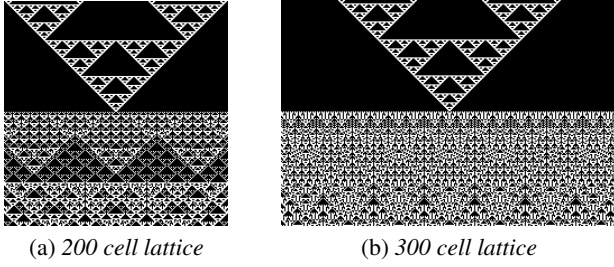


Figure 16: Even steps of 200-step *M54* trajectory on two lattices of different sizes with the same initial conditions.

drastically different state-space trajectories.

In contrast to the many techniques for counting and calculating preimages of CA configurations (Jen, 1989; Voorhees, 1993; Jeras and Dobnikar, 2007; Powley and Stepney, 2010), the reverse trajectory of Rule 90 in Figure 15 is generated from the selfsame process that produces its inverse. This itself is an unexpected result that hints at a deeper significance. Yet there is a further relationship connecting the forward and reverse direction, arising from reflexive composition's symmetric treatment of *state* and *message*: as shown in Figure 17, “transposing” *M45* along the state-message axis produces *M54*, and vice versa. This transposition corresponds to treating messages (inputs and outputs) as states, and states (source and destination nodes) as messages.

What this implies is that the direction of time, for these machines, is fundamentally a product of which elements in the system one chooses to treat as operator and operand. The fact that the two most interesting machines in the set of elementary FSMs exhibit this symmetry hints at the importance of reflexivity in the formulation, and in the unusual dynamics it makes possible. Further investigation of this subtle property and its relationship with complexity is merited.

Conclusions

We have shown that the process referred to as *reflexive composition* generates a surprising diversity of dynamic behavior from simple origins. In particular, the FSMs identified as *M54* and *M60* reverse the state transition trajectory of CA Rule 90 for configurations that have preimages. Unlike techniques developed specifically to uncover such preimages, here they arise naturally out of the formulation, and indeed in the case of *M45* and *M54* are in fact related to each other through a transposition of state and message.

While Rule 90 is among the most famous and well-studied ECA, state-space trajectories explored in earlier research focus either on familiar patterns and their superpositions or on

While not shown here, *M60* transposes into itself and not *M45*. Other machines, notably the billiard ball machine *M44*, do not exhibit this property of state/message transposal resulting in time reversal.

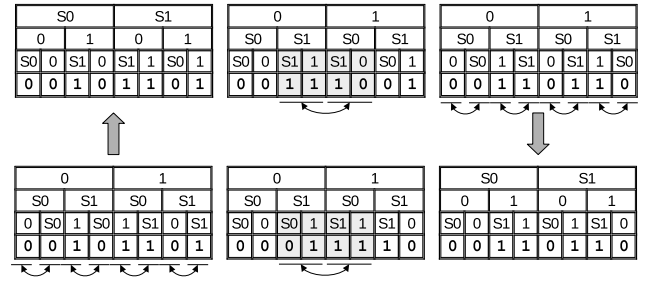


Figure 17: Transposition of state and message transforming *M45* (top left) into *M54* (bottom right), with $S0 \leftrightarrow 0$, $S1 \leftrightarrow 1$. Moving clockwise, the first step swaps state/message in source and input, the second in destination and output.

random initial conditions. In contrast, the reversal of Rule 90 reveals qualitatively distinct patterns containing elements of nested fractal structure of a kind not seen before. We consider this a sign that the underlying formulation presented here, and the origins from which it was developed, manifest deeper truths about the symmetry of information processors and carriers. Future work will expand the scope of exploration to more complex machines and their interactions, with the aim of uncovering these truths and the secrets they hold.

References

- Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363.
- Fontana, W. (1991). Algorithmic chemistry. In C.G. Langton, C. Taylor, J. F. and Rasmussen, S., editors, *Artificial Life II: SFI Studies in the Sciences of Complexity*, vol. X. Addison-Wesley.
- Fontana, W. and Buss, L. W. (1994). “The arrival of the fittest”: Toward a theory of biological organization. *Bulletin of Mathematical Biology*, 56(1):1–64.
- Jen, E. (1989). Enumeration of preimages in cellular automata. *Complex Syst.*, 3.
- Jeras, I. and Dobnikar, A. (2007). Algorithms for computing preimages of cellular automata configurations. *Physica D: Nonlinear Phenomena*, 233(2):95–111.
- Martin, O., Odlyzko, A. M., and Wolfram, S. (1984). Algebraic properties of cellular automata. *Communications in Mathematical Physics*, 93(2):219 – 258.
- Minsky, M. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall.
- Powley, E. J. and Stepney, S. (2010). Counting preimages of homogeneous configurations in 1-dimensional cellular automata. *Journal of Cellular Automata*, 5(4-5):353–381.
- Salzberg, C. (2006a). Complexity scaling of a minimal functional chemistry. In Luis Mateus Rocha, Larry S. Yaeger, M. A. B. and Floreano, D., editors, *Artificial Life X*, pages 165–171. MIT Press, Cambridge, MA.

- Salzberg, C. (2006b). From machine and tape to structure and function: Formulation of a reflexively computing system. *Artificial Life*, 12:487–512.
- Salzberg, C. (2007). A graph-based reflexive artificial chemistry. *BioSystems*, 87:1–12.
- Sayama, H. and Nehaniv, C. L. (2025). Self-reproduction and evolution in cellular automata: 25 years after evoloops. *Artificial Life*, 31(1):81–95.
- Toffoli, T. and Margolus, N. H. (1990). Invertible cellular automata: A review. *Physica D*, 45:229–253.
- Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265.
- Voorhees, B. (1993). Predecessors of cellular automata states i. additive automata. *Physica D: Nonlinear Phenomena*, 68(2):283–292.
- Wills, P. R. (1989). Genetic information and the determination of functional organization in biological systems. *Systems Research*, 6:219–226.
- Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.