

Fast Simulation of Cellular Automata by Self-Composition

Joseph Natal

Institute for Beam Physics and Technology

Karlsruhe Institute of Technology

Hermann-Von-Helmholtz-Platz 1

Eggenstein-Leopoldshafen, Baden-Wuerttemberg 76344, Germany

Oleksiy Al-saadi

Department of Computer Science, Sonoma State University

1801 East Cotati Avenue

Rohnert Park, CA 94928, United States

Computing the configuration of any one-dimensional cellular automaton at generation n can be accelerated by constructing and running a composite rule with a radius proportional to $\log n$. The new automaton is the original one, but with its local rule function composed with itself. Consequently, the asymptotic time complexity to compute the configuration of generation n is reduced from $O(n^2)$ -time to $O(n^2 / \log n)$ but with $O(n^2 / (\log n)^3)$ -space, demonstrating a time-memory tradeoff. Experimental results are given in the case of rule 30.

Keywords: cellular automata; time complexity; rule 30; nonlinear systems

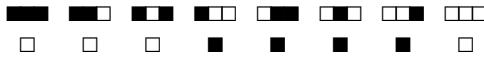
1. Introduction

Compositions of cellular automata (CAs) have been discussed in different contexts since their inception decades ago [1], but their relevance for time complexity improvements has yet to be determined using formal asymptotic analysis. Indeed, compositions were explored by Israeli and Goldenfeld [2] as so-called “coarse-graining,” and were shown to have implications for understanding emergent phenomena in complex systems. Riedel and Zenil [3, 4] further explored coarse-graining, finding subsets of elementary cellular automata (ECAs) that emulate all others, giving evidence of a pervasive universality. Perhaps the most similar algorithm in concept to the one described herein is given by Gosper’s hashed implementation of the Game of Life [5]. This algorithm memorizes local recurrent spacetime patterns for Life-like automata, but performs poorly for chaotic structures. Our focus on rule 30 in this paper stems from its notoriously chaotic nature and perceived lack of structure. The naive method of computing the

configuration at generation n of an elementary cellular automaton (ECA) is simply to update row by row, requiring $O(n^2)$ -time. In contrast, CAs with nested patterns such as the well-studied rules 90 or 150 contain a fractal structure that allows for a given spacetime coordinate to be determined in $O(\log n)$ [6].

2. Preliminaries

A *cellular automaton* (CA) is composed of a regular grid of colored cells whose colors are updated according to a certain rule that may be described by a *rule icon*. For example, the complete rule icon of the well-studied rule 30 CA is:



The rule icon is applied to a row of cells in a one-dimensional CA by iterating through every cell in the row, comparing the colors of a cell along with its two adjacent neighbors, and then updating the color of the cell in the succeeding row (see Figure 1). Let $\mathcal{F}$ be a CA with grid colors indexed by the integers $\mathbb{Z}_k = \{0, 1, \dots, k-1\}$, a given initial state and local update rule $f: \mathbb{Z}_k^{2r+1} \rightarrow \mathbb{Z}_k$. Here, r is a positive integer that denotes the *radius* of the CA: the number of neighbor cells to the left and right of the current cell that are taken into account when applying the rule icon. These cells are sometimes referred to as a *neighborhood*. For example, the rule icon of rule 30 shows that the rule has radius 1 (see the above rule icon).

We will restrict our study of CAs to those with two colors by defining $\Sigma = \mathbb{Z}_2 = \{0, 1\} = \{\square, \blacksquare\}$. A CA will subsequently and informally refer to one that is one dimensional and two color. If a two-color CA $\mathcal{F}$ has radius 1, then we call $\mathcal{F}$ an *elementary cellular automaton* (ECA). Wolfram popularized a systematic numbering scheme for ECA where for any given rule k , the corresponding rule icon can be obtained by taking the binary digits of integer k and assigning them as outputs to the set of binary three-tuples: the numbers 0 through 7 in decimal. Using the above example rule icon, the output cells are, sequentially, 00 011 110₂, which is 30₁₀. There are $2^{2^3} = 256$ ECAs, though many are mirrored and therefore exhibit the same behavior.

A *configuration* $X = \langle x_{-(N-1)/2}, \dots, x_{(N-1)/2} \rangle$ refers to a row of cells having $N \in 2\mathbb{N} + 1$ indices. It will be convention that the median element in the array has index $i = 0$. When necessary to distinguish, $X_n^{\mathcal{F}}$ denotes the bi-infinite configuration of $\mathcal{F}$ at generation n (i.e., after $n - 1$ applications of its local rule beyond the specified initial configuration). It follows that $X_n^{\mathcal{F}} = \langle \dots, x_{i-2}^n, x_{i-1}^n, x_i^n, x_{i+1}^n, x_{i+2}^n, \dots \rangle$. We

explicitly define the *global update* of a configuration as

$$X_{n+1}^{\mathcal{F}} = \bigoplus_{i=-\infty}^{\infty} f \left(\bigoplus_{j=-r}^r x_{i+j}^n \right) \quad (1)$$

and for an individual cell in $X_{n+1}^{\mathcal{F}}$

$$x_i^{n+1} = f \left(\bigoplus_{j=-r}^r x_{i+j}^n \right) \quad (2)$$

where $\oplus$ denotes concatenation of cells. Borrowing from physics terminology, n reflects a dimension in time, while spatial dimension is reflected by the i^{th} index of $X_n^{\mathcal{F}}$. A CA $\mathcal{F}$ has a *simple seed* if $X_1^{\mathcal{F}} = \langle \cdots \square, \square, \square, \blacksquare, \square, \square, \square, \cdots \rangle$.

The rule icons of any CA with radius r can be equivalently represented through an algebraic form as a function $f: \Sigma^{2r+1} \rightarrow \Sigma$. For example, for any generation $n+1$ and index i , the color x_i^{n+1} of rule 150 can be expressly computed by the following trivariate function:

$$x_i^{n+1} = x_{i-1}^n + x_i^n + x_{i+1}^n \bmod 2. \quad (3)$$

The rule 150 CA is well known for its fractal self-symmetry, and its algebraic form is similarly straightforward. Meanwhile, the following nonlinear discrete dynamical equation governs rule 30:

$$x_i^{n+1} = \overbrace{x_{i-1}^n + x_i^n + x_{i+1}^n}^{\text{rule 150}} + \overbrace{x_i^n x_{i+1}^n}^{\text{nonlinear term}} \bmod 2 \quad (4)$$

or equivalently, using Boolean algebra:

$$x_i^{n+1} = x_{i-1}^n \oplus (x_i^n \vee x_{i+1}^n). \quad (5)$$

Notice that the rule function of rule 30 contains the algebraic form of rule 150. Although this component of the function is rather simple in and of itself, the abrupt complexity of rule 30 arises from the nonlinear term. We introduce a special class of graphs that will be used for algorithmic analysis in Lemmas 1–4.

Definition 1. A de Bruijn graph is a directed graph representing overlaps between sequences of symbols. For a two-color CA $\mathcal{H}$ of radius r and local rule h , its de Bruijn graph $B^{\mathcal{H}}$ will have $|\Sigma^{2r}| = 2^{2r}$ vertices, each representing a cell neighborhood of length $2r$. The vertices each have two outgoing edges corresponding to the color of cell $(2r+1)$. These edges are directed to the vertices that represent the neighborhood realized after a unit shift of the original neighborhood. We associate a color with every edge that corresponds to the output of the rule function h applied to the traversed neighborhood.

The de Bruijn graphs have been well studied by Wolfram [1] in the context of cellular automata (CAs). Figure A.1 details the de Bruijn

graph of rule 30, and Figure A.3 shows that these graphs have a very regular structure. These graphs are used for DNA sequence assembly, and so optimization and compression of them is an active area of research [7].

3. Automata Self-Composition

In this section, we show the local rule function of a CA can be composed with itself in order to create a new CA satisfying special constraints on its configurations.

Definition 2. Let the composition of two rules with functions f_1 and f_2 be defined as

$$(f_2 \circ f_1)(X) = f_2 \left(\bigoplus_{i=-r_2}^{r_2} f_1 \left(\bigoplus_{j=-r_1}^{r_1} x_{i+j} \right) \right). \quad (6)$$

This composite rule is therefore a local rule with radius $r_1 + r_2$ and a function mapping $X \in \Sigma^{2(r_1+r_2)+1} \rightarrow \Sigma$.

By Definition 2, the composite rule $f_2 \circ f_1$ consists of applying f_1 to all contiguous subarrays of length $2r_1 + 1$ in X with truncated boundary conditions in indexed order and then f_2 to the concatenated results. In short, the configuration is updated according to f_1 , then f_2 .

Lemma 1 is vital to the correctness of our algorithm.

Lemma 1. Given a CA $\mathcal{H}$ with local rule h and radius r , there exists a CA $\mathcal{G}$ with local rule g and radius $2r$ such that for every $n \in \mathbb{N}$ we have that $X_{2n-1}^{\mathcal{H}} = X_n^{\mathcal{G}}$.

Proof. For self-composition, equation (6) is reduced to

$$(h \circ h)(X) : \left\{ \Sigma^{4r+1} \rightarrow \Sigma : X \mapsto h \left(\bigoplus_{i=-r}^r h \left(\bigoplus_{j=-r}^r x_{i+j} \right) \right) \right\}. \quad (7)$$

Replacing h with $h \circ h$ in the global update (equation (1)) and assigning a generation n gives

$$X_{n+1}^{\mathcal{G}} = \bigoplus_{i=-\infty}^{\infty} (h \circ h) \left(\bigoplus_{j=-2r}^{2r} x_{i+j}^n \right) = \bigoplus_{i=-\infty}^{\infty} h \left(\bigoplus_{j=-r}^r h \left(\bigoplus_{k=-r}^r x_{i+j+k}^n \right) \right), \quad (8)$$

and substituting $x_{i+j}^{n+1} = h \left(\bigoplus_{k=-r}^r x_{i+j+k}^n \right)$ from equation (2)

$$= \bigoplus_{i=-\infty}^{\infty} h \left(\bigoplus_{j=-r}^r x_{i+j}^{n+1} \right) = X_{n+2}^{\mathcal{H}}. \quad (9)$$

Then the composite update is simply two global updates according to h in sequence. Hence, we construct $\mathcal{G}$ as follows: Let $g = h \circ h$ be the local rule function of $\mathcal{G}$ with the same initial configuration as $\mathcal{H}$ (i.e.,

$X_1^{\mathcal{H}} = X_1^{\mathcal{G}}$). Then we have that $X_3^{\mathcal{H}} = X_2^{\mathcal{G}}$ by equation (9). It follows that by repeated application of the global update for $\mathcal{G}$ for $n = 1, 2, 3, \dots$, the evolution for $\mathcal{H}$ will have $1, 3, 5, 7, \dots = 2n - 1$ and $X_{2n-1}^{\mathcal{H}} = X_n^{\mathcal{G}}$. Intuitively, running $\mathcal{G}$ for a single generation provides the same output configuration as running $\mathcal{H}$ for two generations (Figure 1). $\square$

k -fold of rule 30	$2r + 1$	Rule
1	3	30
2	5	535 945 230
3	7	4 245 223...81 390
4	9	1 672 702...88 750

Table 1. Several rules are shown that can be constructed through self-composition of rule 30, along with their radii.

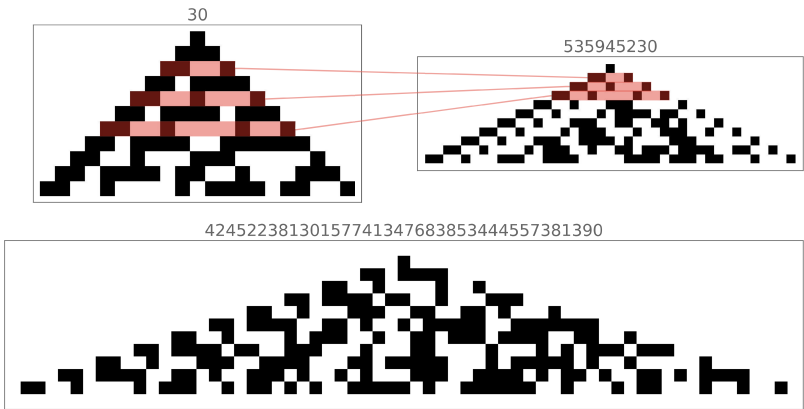


Figure 1. Evolution of rule 30 beginning with a simple seed and its two-fold and three-fold composition with itself (see Definition 3). Highlighted rows show a sample of equivalent configurations.

Now we extend this composition of automata to arbitrary k and define a new terminology to describe CA that are of this type:

Definition 3. Given a CA $\mathcal{H}$ with local rule h and a CA $\mathcal{F}$ with local rule f , let $h_i = h$ for each $1 \leq i$. We say that $\mathcal{F}$ is a k -fold composition of $\mathcal{H}$ if $f = h_1 \circ h_2 \circ \dots \circ h_{k-1} \circ h_k$. Equivalently, we write $f = h^{(k)}$.

It follows by the conclusion of Lemma 1 that $\mathcal{G}$ is a two-fold composition of $\mathcal{H}$. It is obvious self-composition is associative because the composing functions are identical by definition.

Lemma 2. Given a CA $\mathcal{F}$ that is a k -fold composition of $\mathcal{H}$, if $X_1^{\mathcal{F}} = X_1^{\mathcal{H}}$ then $X_{kn-1}^{\mathcal{H}} = X_n^{\mathcal{F}}$. The composite local rule f has radius kr .

Proof. The k -fold composition of h is

$$h^{(k)}(X) : X \mapsto h \left(\bigoplus_{\alpha_1=-r}^r h \left(\bigoplus_{\alpha_2=-r}^r \dots h \left(\bigoplus_{\alpha_k=-r}^r x_{\alpha_1+\alpha_2+\dots+\alpha_k} \right) \right) \right) \quad (10)$$

and maps $X \in \Sigma^{2(kr)+1} \rightarrow \Sigma$. The proof is given by induction on repeated application of Lemma 1. Instead of two updates ($k = 2$), there are k configuration updates per input configuration, corresponding to the number of nested applications of h by equation (10). $\square$

Lemma 3 shows that a k -fold composition of a CA can be quickly computed.

Lemma 3. Given a CA $\mathcal{H}$, the k -fold composition $\mathcal{F}$ can be computed in $O(k^2 2^{2kr})$ -time. Moreover, $B^{\mathcal{F}}$ can be constructed with no additional runtime complexity.

Proof. Note that r , the initial rule radius, is constant. It is the time complexity scaling with k that is to be resolved. By Lemma 2, $h^{(k)}$ has radius kr . In order to compute $h^{(k)}$, the local rule that governs $\mathcal{F}$, there are $|\Sigma^{2kr+1}| = 2^{2kr+1}$ configurations (i.e., $(2kr+1)$ -tuples) that need to be iterated over. Each configuration requires $O(k^2)$ -time to compute (see Example 1).

The corresponding de Bruijn graph $B^{\mathcal{F}} = (V, E)$ is straightforward to construct. We have that $|V| = 2^{2r}$ vertices. By Definition 1, the de Bruijn graph $B^{\mathcal{F}}$ (see Figure A.3) has outgoing edges

$$E = \{v_i \in V \mid v_i \rightarrow (v_{2i \bmod |V|}, v_{(2i+1) \bmod |V|})\},$$

where index i in binary is equal to the corresponding cell configuration in Σ^{2r} . The color of each edge is determined by the computation in the first half of this lemma. $\square$

Example 1. We illustrate the time complexity with a concrete example for $r = 1$, $k = 3$ and h as rule 30. This is a three-fold composition of rule 30. Since we have $2(kr) + 1 = 7$, there are 2^7 possible input configurations. Let $X = \langle \square, \square, \blacksquare, \blacksquare, \square, \blacksquare, \square \rangle \in \Sigma^7$. Applying the rule h once to X in accordance with Definition 2 gives $\langle \blacksquare, \blacksquare, \square, \square, \blacksquare \rangle$. Applying h twice more gives $\langle \square, \blacksquare, \blacksquare \rangle$ and then $\blacksquare$. So $h^{(3)}(X) = \blacksquare$. Clearly, there is an $O(k^2)$ array of cells that have been updated.

The use of a de Bruijn graph will provide sufficient algorithmic analysis for running a k -fold composition.

Lemma 4. Let $\mathcal{H}$ be a CA having a local rule h and radius r . Let $X_n^{\mathcal{H}}$ be the current configuration and let $B^{\mathcal{H}}$ be the de Bruijn graph of $\mathcal{H}$. Then, $X_{n+1}^{\mathcal{H}}$ can be computed in $O(n)$ -time.

Proof. To compute $X_{n+1}^{\mathcal{H}}$, a walk is made on the graph $B^{\mathcal{H}}$ (see Figure A.2 for an example where $\mathcal{H}$ is rule 30). Each edge traversed is constant time, where the edge chosen is based on the color of the next cell in the configuration. For any finite configuration length, the walk will begin and end at $\{0\}^{2r}$. Because $X_n^{\mathcal{H}}$ has size $O(n)$, iterating through each color in the configuration to compute $X_{n+1}^{\mathcal{H}}$ requires $O(n)$ -time. $\square$

We are prepared to complete our main result:

Theorem 1. Let $\mathcal{H}$ be a CA with an arbitrary initial configuration, a local rule h and radius r . The configuration $X_n^{\mathcal{H}}$ can be computed in $O(n^2 / \log n)$ -time.

Proof. By Lemma 3, we require $O(k^2 2^{2kr})$ -time to generate the k -fold composition $h^{(k)}$ of $\mathcal{H}$. Let $\mathcal{F}$ be the CA governed by the local rule $h^{(k)}$, and by the same lemma we construct $B^{\mathcal{F}}$. By Lemma 4, we can compute the next generation of any configuration of $\mathcal{F}$ in $O(n)$.

From a simple geometric argument, it follows that the k -fold composition $h^{(k)}$ reduces the time complexity from $O(n^2)$ to $O(n^2 / k)$: for each row that is computed, k are skipped relative to the original rule. Then there exists an optimal maximum k such that the time complexity to run the simulation from the initial configuration to n and compose the rule is equivalent:

$$\frac{n^2}{k} = k^2 2^{2kr} \quad (11)$$

up to an arbitrary constant. It is assumed that n, k are much larger than r , the initial rule radius. If the time complexities are equal, the total complexity is the complexity of either operation multiplied by a constant factor. The solution to the equation is then the Lambert W function (also called the “product logarithm”)

$$k(n) = \frac{3}{2r \ln(2)} W_0\left(\frac{2r \ln(2)}{3} n^{2/3}\right) \sim \log n \quad (12)$$

by using $W_0(n) \sim \log n$ for large n and simplifying. Thus it is shown that the time complexity is dominated by the exponential number of states for a given k . In fact, the prefactor k^2 in the cost to generate the composite rule output can be any positive polynomial and not change the asymptotic time complexity. $\square$

Corollary 1. The configuration $X_n^{\mathcal{H}}$ satisfying the conclusion of Theorem 1 can be computed using $O(n^2 / (\log n)^3)$ -space.

Proof. Rearranging equation (11) to solve for the number of states $|\Sigma^{2kr+1}| = 2^{2kr+1} \sim n^2 / k^3$ and in the asymptotic limit of large n , we arrive at $\sim n^2 / (\log n)^3$. $\square$

4. Experimental Results

In this section, we present experimental results of the method described in Theorem 1, managing to remove a log factor from the base runtime complexity of trivially running an ECA. The results are presented in Figures 2 through 4. The practicality of implementing the code on a general-purpose computer is limited by two things: (1) memory access is not sequential and so is not cache friendly; (2) random-access memory (RAM) is limited. As it turns out, the bitwise optimizations described by Wolfram [8] are several times faster than this method for any reasonable number of generations because computers parallelize packed bitwise operations and are efficient at reading memory linearly along an array.

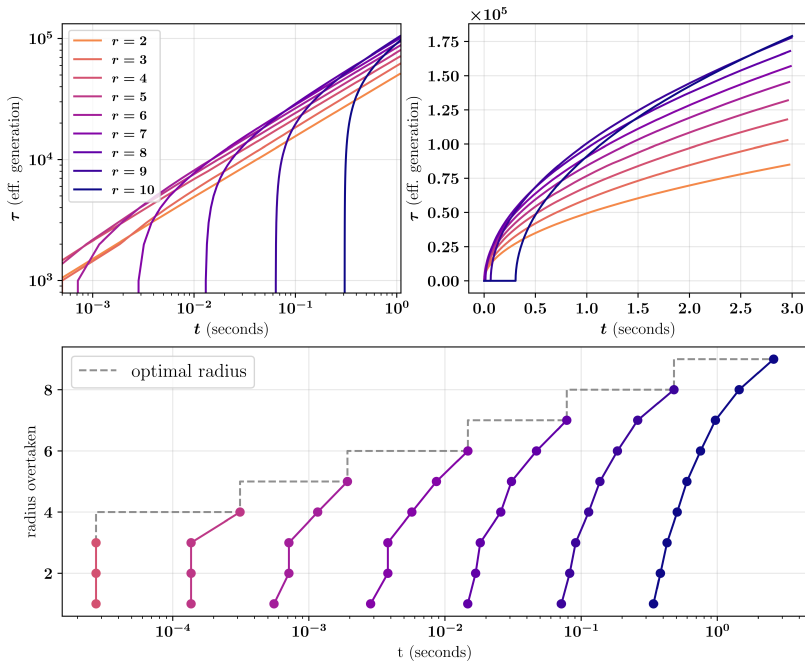


Figure 2. Constructing a larger-radius automaton improves simulation speed on a given machine (Intel® Xeon® Gold 6230 CPU @ 2.10GHz). The bottom panel shows the optimal radius as a function of time. It also shows the times at which a simulation at a given radius surpasses those of a smaller radius given on the vertical axis. Given a fixed amount of simulation time t seconds, the optimal radius is approximated by $\lfloor 0.37 \log_2(t) + 9.4 \rfloor$.

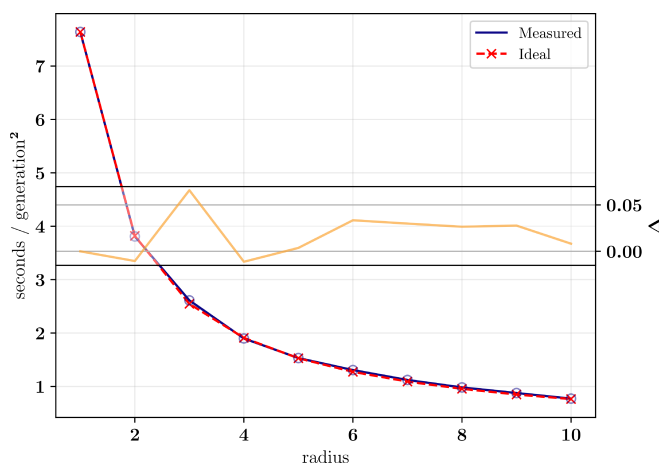


Figure 3. The machine in this experiment obeys an n^2/k time complexity scaling law ($k = r$) to compute the next generation, validating the use of equation (11). The ideal curve uses $r = 1$ as a reference, so if $f(r)$ is the measured number of seconds per squared generation, the ideal is $f_{\text{ideal}}(r) = f(r = 1)/r$.

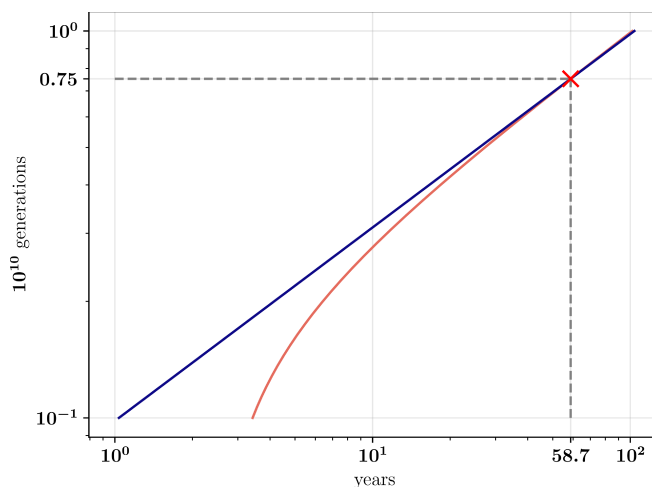


Figure 4. The 27-fold composition (4.5 petabytes) will overtake the bitwise-optimized implementation in about 60 years on an Intel® Xeon® Gold CPU. This takes the principle of delayed gratification to its extreme. This algorithm might only be practical with special hardware. The curves were generated by extrapolating the quadratic time versus generation curves and the exponential dependence on r for composite rule creation.

These memory problems could be reduced if contiguous memory is indexed in a de Bruijn sequence [9] of order $2r + 1$ on the alphabet Σ . This is because, given a rule input $\langle \blacksquare, \square, \blacksquare \rangle$, the adjacent one in the configuration is either $\langle \square, \blacksquare, \blacksquare \rangle$ or $\langle \square, \blacksquare, \square \rangle$ which is a bit shift left and rewriting of the rightmost bit (flipped depending on the endianness). If the index in memory is q , then the next transition will map to memory location $\lfloor q/2 \rfloor$ or $\lfloor q/2 \rfloor + 2^{2r}$, which may be a large jump. For two-color CAs with an equal probability of being black or white, there would be an equal chance that the next state would be in the neighboring memory address.

Nonetheless, Figure 3 shows that computers access memory at speeds independent of address size in the plotted regime. And despite the aforementioned memory inefficiencies, creating large compositions would eventually overtake the bitwise-optimized implementation, as shown in Figure 4.

5. Discussion

This result is a single point on the domain of spacetime functional dependencies that are possible for computing an arbitrary generation of an elementary cellular automaton (ECA), and its method is straightforward. On a computer, it relies on a model of computation that treats accessing memory with an address of size $O(\log n)$ as taking a single unit of time—the canonical RAM model. This model is a convention brought about by advances in physical machinery [10]. One might argue that a proper machine must be a Boolean circuit that uses the standard basis of Boolean operations. But in this computational model of circuit complexity, the interpretation of our result is clear: our algorithm increases circuit size while decreasing circuit depth in accordance with our equations. The operation of traversing an edge on the de Bruijn graph is realizable in constant circuit depth, which is what we would define as time in that model. With the contemporary interest in interaction nets, graph rewriting and category theory, perhaps a different model of computation will become standard.

There may exist faster machines to compute certain cellular automata (CAs), and maybe some that can be described by a simple equation over the integers. Open questions remain about the possibility of such formal machines, their potential for practical instantiation and their implications for emergent phenomena in complex systems.

Appendix

Figures A.1 through A.3 show several de Bruijn graph examples. Available source code can be compiled using the C++17 standard.

Alternatively, the functions are implemented in a few lines of compiled Wolfram Language code. See github.com/Joseph459459/automata-compositions.

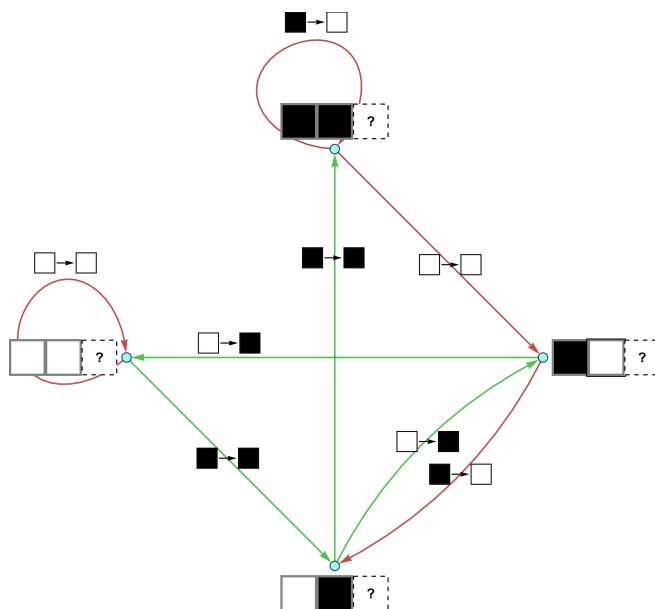


Figure A.1. Rule 30's state transition (colored de Bruijn) diagram. The left cell in the edge rule $\{\square, \blacksquare\} \rightarrow \{\square, \blacksquare\}$ is read in *from* the cell configuration, and the right cell is written *to* the configuration. Red edges visually indicate this output cell is $\blacksquare$ and green edges indicate the output is $\square$. As an edge is traversed, the neighborhood is shifted left by one cell.

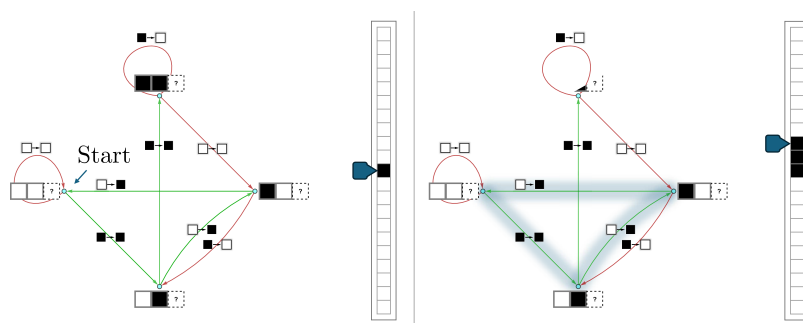


Figure A.2. Computing the next generation of the ECA rule 30 is taking a walk in the state transition graph. The figure on the right shows the walk taken in the graph in order to produce $\langle \blacksquare, \blacksquare, \blacksquare \rangle$.

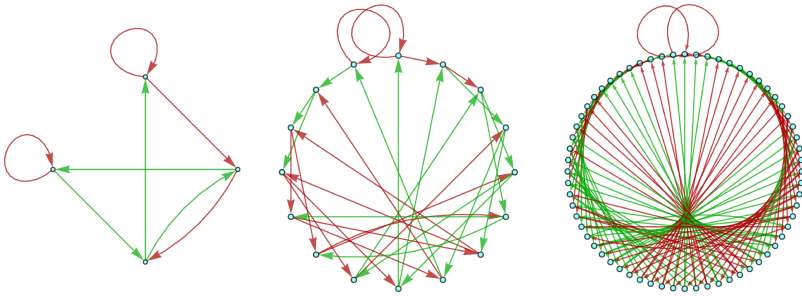


Figure A.3. One-fold, two-fold and three-fold composition state transition diagrams of the rule 30 ECA in circular embeddings [11]. They have a regular structure, with any vertex at index i along the circular embedding having outgoing connections to vertices at $2i$ and $2i + 1$ modulo the number of vertices.

References

- [1] S. Wolfram, “Computation Theory of Cellular Automata,” *Communications in Mathematical Physics*, **96**(1), 1984 pp. 15–57. doi:10.1007/BF01217347.
- [2] N. Israeli and N. Goldenfeld, “Coarse-Graining of Cellular Automata, Emergence, and the Predictability of Complex Systems,” *Physical Review E*, **73**(2), 2006 026203, doi:10.1103/PhysRevE.73.026203.
- [3] J. Riedel and H. Zenil, “Cross-Boundary Behavioural Reprogrammability Reveals Evidence of Pervasive Universality,” *International Journal of Unconventional Computing*, **13**(4–5), 2017–2018 pp. 309–357. www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-13-number-4-5-2018/ijuc-13-4-5-p-309-357.
- [4] J. Riedel and H. Zenil, “Rule Primality, Minimal Generating Sets and Turing-Universality in the Causal Decomposition of Elementary Cellular Automata,” *Journal of Cellular Automata*, **13**(5–6), 2018 pp. 479–497. www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-13-number-5-6-2018/jca-13-5-6-p-479-497.
- [5] R. Wm. Gosper, “Exploiting Regularities in Large Cellular Spaces,” *Physica D: Nonlinear Phenomena*, **10**(1), 1984 pp. 75–80, doi:10.1016/0167-2789(84)90251-3.
- [6] K. Culik II and S. Dube, “Fractal and Recurrent Behavior of Cellular Automata,” *Complex Systems*, **3**(3), 1989 pp. 253–267. complex-systems.com/pdf/03-3-3.pdf.
- [7] M. D. Muggli, A. Bowe, N. R. Noyes, P. S. Morley, K. E. Belk, R. Raymond, T. Gagie, S. J. Puglisi and C. Boucher, “Succinct Colored de Bruijn Graphs,” *Bioinformatics*, **33**(20), 2017 pp. 3181–3187, doi:10.1093/bioinformatics/btx067.

- [8] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002. www.wolframscience.com.
- [9] E. W. Weisstein. “de Bruijn Sequence” from Wolfram MathWorld—A Wolfram Web Resource.
mathworld.wolfram.com/deBruijnSequence.html.
- [10] D. E. Knuth, “The Dangers of Computer-Science Theory,” in *Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science*, Bucharest (P. Suppes, L. Henkin, A. Joja and G. C. Moisil, eds.), pp. 189–195, Amsterdam: Elsevier, 1973 pp. 189–195. doi:10.1016/S0049-237X(09)70357-X.
- [11] K. Sutner, “Automata Theory Package” from the Wolfram Paclet Repository—A Wolfram Web Resource.
resources.wolframcloud.com/PacletRepository/resources/KlausSutner/Automata.