# Measuring Behavioral Similarity of Cellular Automata

Peter D. Turney*
Ronin Institute
peter.turney@ronininstitute.org

**Keywords**
Cellular automata, semi-totalistic automata, outer-totalistic automata, Conway's Game of Life, similarity of cellular automata, behavior of cellular automata

**Abstract** Conway's Game of Life is the best-known cellular automaton. It is a classic model of emergence and self-organization, it is Turing-complete, and it can simulate a universal constructor. The Game of Life belongs to the set of semi-totalistic cellular automata, a family with 262,144 members. Many of these automata may deserve as much attention as the Game of Life, if not more. The challenge we address here is to provide a structure for organizing this large family, to make it easier to find interesting automata, and to understand the relations between automata. Packard and Wolfram (1985) divided the family into four classes, based on the observed behaviors of the rules. Eppstein (2010) proposed an alternative four-class system, based on the forms of the rules. Instead of a class-based organization, we propose a continuous high-dimensional vector space, where each automaton is represented by a point in the space. The distance between two automata in this space corresponds to the differences in their behavioral characteristics. Nearest neighbors in the space have similar behaviors. This space should make it easier for researchers to see the structure of the family of semi-totalistic rules and to find the hidden gems in the family.

## 1 Introduction

The *Game of Life* (GoL) is a solitaire game invented by John Conway and introduced to the world by Martin Gardner (1970) in *Scientific American*. It is played on a potentially infinite, two-dimensional grid of square cells. Each cell is either *dead* (state 0) or *alive* (state 1). The state of a cell changes with time, based on the states of its eight nearest neighbors (called the *Moore neighborhood*). Time passes in a series of discrete intervals. At time $t = 0$, the player of the game chooses the initial states of the grid. The initial states form a *seed pattern* that determines the course of the game. The states at time $t$ uniquely determine the states at time $t + 1$. With each increment of $t$, all of the cells are updated. As the game runs, patterns grow and decay, resembling living organisms.

The rule for changing states in GoL can be compactly represented as B3/S23, where *B* means "born" and *S* means "survives." A cell is *born* (it switches from state 0 to state 1) when exactly three of its eight nearest neighbors are alive (in state 1). A cell *survives* (remains in state 1) when it has either two or three living neighbors. Otherwise, a cell *dies* (it switches to state 0 or remains in state 0).

GoL is a *cellular automaton*, a discrete, abstract computational system. It is the best-known member of the family of cellular automata. It is popular as a model of emergence and self-organization

* Corresponding author.

© 2021 Massachusetts Institute of Technology.
Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

(Bak et al., 1989), it is Turing-complete (Rendell, 2016), and it is a universal constructor (LifeWiki, 2021b).

GoL is a member of the family of *semi-totalistic cellular automata* (also called "outer-totalistic"). The rules for this family have the general form B$x$/S$y$, where $x$ and $y$ are generated by deleting digits from the string 012345678, including deleting no digits or deleting all digits (Eppstein, 2010). Since there are nine digits available for B and nine digits available for S, there are 2 to the power of 18 (262,144) possible semi-totalistic rules.

Packard and Wolfram (1985) introduced a four-class system for characterizing two-dimensional cellular automata. However, as Eppstein (2010, p. 72) reports, "The boundary between classes is less clear-cut and more subjective than one would like, and may for some automata be impossible to decide." Eppstein (2010) proposed an alternative four-class system, based on an analysis of the B$x$/S$y$ rule forms. We will compare Eppstein's four-class system with our approach in section 3.

In this article, we introduce a continuous high-dimensional vector space, where the statistical behavior of an automaton corresponds to a point in the space. For a given semi-totalistic rule, such as B3/S23, we create an initial random soup of living cells and then run the game. We then randomly sample cells in the game to estimate the probabilities of the state transitions, which depend on both the rules of the game and the particular patterns of neighbors that tend to arise in the game. The behavior is then represented as a vector of the estimated probabilities of each possible state transition.

With the probability vectors for all 262,144 semi-totalistic cellular automata, we can measure the behavioral similarity of any two automata by any suitable distance measure. In this article, we have chosen to use Euclidean distance, where low distance corresponds to high behavioral similarity. This continuous high-dimensional vector space enables many ways of searching for interesting automata and finding relations among automata, as we show in Table 1.

The source code for calculating the high-dimensional vectors is available for downloading, along with the vectors for all 262,144 possible semi-totalistic rules (Turney, 2021). The source code uses Python and Golly (Trevorrow et al., 2021). In section 2, we describe the problems of *strobing* and *infinity* that arise with rules that contain B0. In section 3, we present a 36-dimensional space for representing the behavior of automata, ignoring the issues of strobing and infinity, in order to simplify the presentation. In section 4, we address the problems of strobing and infinity, using a 72-dimensional space. We conclude in section 5.

## 2 The Problems of Strobing and Infinity

In a typical game, we begin at time $t = 0$ with a potentially infinite grid and a finite number of live cells. If the given rule contains B0, all dead cells with no neighbors will come alive at time $t = 1$. This means there will be an infinite number of live cells at $t = 1$. If the rule does not contain S8, at time $t = 2$, an infinite number of cells will die. This creates an annoying strobing effect, with alternating light and dark images at odd and even times.

The popular Golly software for cellular automata automatically corrects the problems with strobing and infinity (Golly, 2021). If a rule contains B0 and S8, then the rule is replaced with an equivalent rule that avoids the problem of an infinite number of live cells. If a rule contains B0 but not S8, then the rule is replaced with two rules, one for even times and one for odd times, to avoid the problem of strobing. We do not have the space here to explain how Golly adjusts the rules, but this information is available on the Golly website (Golly, 2021) and our Python code includes the required adjustments (Turney, 2021).

## 3 The 36-Dimensional Space for Semi-Totalistic Automata

As an example of the 36-dimensional space for semi-totalistic automata, Table 2 shows two vectors for the Game of Life, B3/S23. The first is a Boolean vector that expresses the rule B3/S23 with four 9-dimensional vectors, Born (B), Survive (S), Unborn (U), and Die (D). The rule tells us which state transitions are *allowed* (1) and which transitions are *forbidden* (0). We can see that the Boolean vector



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

Table 1. This table lists potential applications for a continuous high-dimensional vector space representation of the characteristics of cellular automata.

| Applications for vectors | Examples |
|---|---|
| Find more like this | Given an automaton that is interesting, find similar automata that might also be interesting, by sorting the automata in order of increasing distance from the given automaton. |
| Finding hybrids | Given two different automata, find a third automaton that is half way between them. |
| Finding models for phenomena | Given some natural phenomenon that looks like it might be something a cellular automaton could model, try to make a behavioral vector for it, then use the behavioral vector to find the best automaton for modeling the phenomenon. |
| Unsupervised clustering | Use a similarity-based clustering algorithm to cluster automata by vector similarity. |
| Supervised clustering | Use vector similarity for supervised clustering, given some manually identified examples of each desired cluster. |
| Finding opposites | Given an automaton, find the automaton that is least similar to the given rule; the maximally distant automaton. |
| Finding idiosyncrasy | For each automaton, find its nearest neighbor, then make note of how different the two automata are; output the automaton that is least like its nearest neighbor (most unique). |
| Finding paradigmatic examples | Given a group of automata that belong to the same cluster, find the automaton that is the centroid or average of the cluster, to serve as a paradigmatic example of the cluster. |
| Projection into subspaces | To understand the high-dimensional vector space better, project it into lower dimensional subspaces, such as two-dimensional subspaces, which are easier to visualize. |

for U is the inverse of the vector for B and D is the inverse of S. A limitation of measuring distance with Boolean vectors is that any two rules that differ by the insertion or deletion of $N$ numbers are exactly the same distance apart. The Boolean vector space is not capable of making fine distinctions between rules.

The second vector is a real-valued vector of probabilities, such that the sum of the probabilities is 1. This vector tells us the estimated probability of each *allowed* state transition, based on many observations of the Game of Life. State transitions that are *forbidden* have a probability of zero. Table 2 shows that a cell with two neighbors is more likely to survive than a cell with three neighbors (probability 0.0414 for two and probability 0.0327 for three). This implies that a line of live cells is slightly more likely to continue than to branch out. The probability vectors give us qualitative behavioral information that is not available in the Boolean vectors.

The probabilities in Table 2 were generated by repeatedly making random soups and running them to see how they evolve. Table 3 shows the parameter settings we used for these experiments. Taking a sample consists of randomly selecting a cell (alive or dead) and inspecting its state and the state of its neighbors at time $t$ and then inspecting its new state at time $t + 1$, to see what transition took place.

Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

Table 2. Here we see two 36-dimensional vectors for the Game of Life, B3/S23.

| Vectors | | Transition $t$ to $t+1$ | Number of neighbors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Boolean | B | 0 to 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| B3/S23 | S | 1 to 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | U | 0 to 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| | D | 1 to 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Real | B | 0 to 1 | 0 | 0 | 0 | 0.0454 | 0 | 0 | 0 | 0 | 0 |
| B3/S23 | S | 1 to 1 | 0 | 0 | 0.0414 | 0.0327 | 0 | 0 | 0 | 0 | 0 |
| | U | 0 to 0 | 0.5883 | 0.0963 | 0.1082 | 0 | 0.0212 | 0.0199 | 0.0037 | 0.0003 | 0.0002 |
| | D | 1 to 0 | 0.0013 | 0.0149 | 0 | 0 | 0.0160 | 0.0073 | 0.0025 | 0.0004 | 0.0001 |

*Note*. The Boolean vector expresses exactly the same information as the rule, B3/S23, but more explicitly. The real-valued probability vector gives a more detailed picture, showing how the rule behaves in practice.

The initial random soup is contained in a 16 × 16 square of cells (initial_size), following the example of Catagolue (LifeWiki, 2021a), the most popular software for exploring random soups. Catagolue is the latest in a series of tools for exploring soups, and it is widely used by the GoL community. Catagolue has been running continuously since 2015, and "at least 18,928,504,510,982 soups have been investigated by the census's participants, yielding a total of at least 413,493,174,300,923 objects of 159,347 distinct types" (LifeWiki, 2021a, para. 2).

The soup is generated in two steps. First, we randomly select a number $d$ using a continuous uniform distribution between 0 and 1 (density_range). This $d$ gives us the desired density of live cells for the 16 × 16 square. Second, we iterate through the cells in the square, randomly assigning state 1 with probability $d$ and state 0 with probability $1 - d$. We then run the soup for 50 steps, to see how it develops (num_steps). We then take 50 samples of the state transitions (num_samples). This process is repeated 1,000 times (num_trials), each time with a different density $d$. From the 50,000 state transitions (1,000 soups with 50 samples per soup), we estimate the state transition probabilities for the real-valued probability vectors.

After 50 steps (num_steps), the random soup may not have settled into a stable configuration, but we believe that some degree of instability is natural, so it is not necessary to run the game until it is

Table 3. This table shows the parameter settings we used for the experiments that estimated the probabilities in the real-valued vectors.

| Parameter | Value | Description |
|---|---|---|
| density_range | [0.0, 1.0] | density range for initial random soup matrix |
| initial_size | 16 | width and height for initial random soup matrix |
| num_steps | 50 | number of steps for a run of the cellular automaton |
| num_samples | 50 | number of samples to collect from each run |
| num_trials | 1,000 | number of trials (runs) to evaluate for each rule |



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

completely stable. It is also advantageous to keep the number of steps relatively small, due to the time required for computation, since we are dealing with 262,144,000 soups (num_trials $\times$ $2^{18}$).

Eppstein (2010) was primarily interested in engineered patterns, designed with a purpose. Therefore, he chose to avoid B0 rules in his analysis of semi-totalistic cellular automata, arguing that B0 rules are not conducive to engineering. Without B0 rules, strobing and infinity are not an issue, so we can compare Eppstein's four-class system with our 36-dimensional space.

A rule is defined as fertile if it has a finite pattern that eventually escapes any bounding box Eppstein (2010). Eppstein defines the following four classes: (1) If a rule includes B1, it is fertile, given a single live cell as a seed pattern. (2) Otherwise, if a rule includes B2, it is fertile, given a $2 \times 2$ block of live cells as a seed pattern. (3) Otherwise, if a rule does not include B1, B2, or B3, it is *not* fertile, because the dead cells outside of the bounding box can have at most three live neighbors. (4) Otherwise, the rule must begin with B3. Some of the B3 rules are fertile and some are not. In some cases, it can be difficult to determine whether B3 rules are fertile or infertile.

Table 4 shows that, if two rules are near neighbors in our 36-dimensional space, then they are almost certainly in the same Eppstein class. This mutual agreement between Eppstein's classes and our 36-dimensional space lends support to both approaches.

To generate Table 4, we randomly chose 10,000 target rules and, for each target rule, we randomly chose another 10,000 candidate rules, from which we picked out the candidate that was nearest (but not identical) to the given target rule. For each target-neighbor pair, we then checked their Eppstein classes. The table shows the percentages for each possible target−neighbor pair of Eppstein's classes.

The numbers on the diagonal in Table 4 tend to decrease as the class numbers increase (50.05, 25.02, 11.26, 12.11). This is because the four classes are defined like a set of four successive sieves, so each subsequent sieve tends to catch fewer cases than its predecessor.

## 4 The 72-Dimensional Space for Semi-Totalistic Automata

As an example of the 72-dimensional space for semi-totalistic automata, Table 5 shows the vectors for the rule B03/S23. This rule causes strobing, because it contains B0 and not S8 (see section 2), so we replace it with two 36-dimensional Boolean vectors, one for even values of $t$ (B1245678/S0145678) and one for odd values of $t$ (B56/S58) (Golly, 2021). To express these anti-strobing rules in a vector space, we need to join these two 36-dimensional Boolean vectors, creating a 72-dimensional Boolean vector. The bottom eight rows of the table show the corresponding 72-dimensional real-valued vector of probabilities. The probabilities in the first 36 dimensions sum to 1, and the probabilities in the second 36 dimensions also sum to 1, so the whole vector sums to 2.

Rules without strobing do not require 72 dimensions, but we can easily expand them to 72-dimensions by repeating the 36-dimensional vector. The 72-dimensional vector has an even part

Table 4. This table shows the relation between Eppstein's (2010) four-class system and our 36-dimensional vector space.

| | Class 1 | Class 2 | Class 3 | Class 4 | Total of classes |
|---|---|---|---|---|---|
| **Class 1** | **50.05** | 0.00 | 0.04 | 0.00 | 50.09 |
| **Class 2** | 0.00 | **25.02** | 0.00 | 0.01 | 25.03 |
| **Class 3** | 0.78 | 0.01 | **11.26** | 0.35 | 12.40 |
| **Class 4** | 0.09 | 0.07 | 0.21 | **12.11** | 12.48 |
| **Total of classes** | 50.92 | 25.10 | 11.51 | 12.47 | 100.00 |

*Note*. All numbers in the table are percentages. The numbers on the diagonal are highlighted in **bold**. The table shows that near neighbors in the 36-dimensional vector space tend to belong to the same class.



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

Table 5. Here we have vectors for the strobing rule B03/S23.

| Vectors | | Transition | Number of neighbors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *t* to *t* + 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Boolean B03/S23 | B | 0 to 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | S | 1 to 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | U | 0 to 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| | D | 1 to 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Boolean B1245678/S0145678 even *t* | B | 0 to 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| | S | 1 to 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | U | 0 to 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | D | 1 to 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Boolean B56/S58 odd *t* | B | 0 to 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| | S | 1 to 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | U | 0 to 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| | D | 1 to 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| Real B1245678/S0145678 even *t* | B | 0 to 1 | 0 | 0.1030 | 0.1332 | 0 | 0.0892 | 0.0489 | 0.0182 | 0.0033 | 0.0004 |
| | S | 1 to 1 | 0.0090 | 0.0371 | 0 | 0 | 0.0628 | 0.0311 | 0.0125 | 0.0040 | 0.0147 |
| | U | 0 to 0 | 0.1562 | 0 | 0 | 0.1277 | 0 | 0 | 0 | 0 | 0 |
| | D | 1 to 0 | 0 | 0 | 0.0703 | 0.0785 | 0 | 0 | 0 | 0 | 0 |
| Real B56/S58 odd *t* | B | 0 to 1 | 0 | 0 | 0 | 0 | 0 | 0.0756 | 0.0622 | 0 | 0 |
| | S | 1 to 1 | 0 | 0 | 0 | 0 | 0 | 0.1309 | 0 | 0 | 0.0408 |
| | U | 0 to 0 | 0.1049 | 0.0189 | 0.0416 | 0.0558 | 0.0652 | 0 | 0 | 0.0303 | 0.0062 |
| | D | 1 to 0 | 0.0001 | 0.0033 | 0.0250 | 0.0690 | 0.0999 | 0 | 0.1045 | 0.0656 | 0 |

*Note*. First, we have the 36-dimensional Boolean vector for B03/S23. Second, we have two 36-dimensional Boolean vectors, one for even *t* and one for odd *t*, which combine to create a 72-dimensional Boolean vector that does not strobe. Third, we have two 36-dimensional probability vectors, which combine to create a 72-dimensional probability vector.

and an odd part, so it is natural to duplicate the 36-dimensional vector when the same rule is used for even and odd times. This allows us to compare non-strobing rules (such as B3/S23) with strobing rules (such as B03/S23) in the same 72-dimensional space.

Table 6 shows the nearest neighbors of the rule B3/S23 in the 72-dimensional probability space. There are two cases where Golly modifies rules, in order to prevent strobing and infinity (Golly,



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

Table 6. This table shows the twenty nearest neighbors of the Game of Life, the target rule B3/S23.

| Rank | Rule | Anti-infinity rule change | Duplicate rules | Distance from B3/S23 | |
|---|---|---|---|---|---|
| | | | | Real | Boolean |
| 1 | B3/S23—target rule | | 1 & 2 | 0.0000 | 0.0000 |
| 2 | B0123478/S01234678 | B3/S23 | 1 & 2 | 0.0146 | 0.0000 |
| 3 | B0123478/S1234678 | B38/S23 | 3 & 5 | 0.0269 | 1.4142 |
| 4 | B3/S238 | | | 0.0298 | 1.4142 |
| 5 | B38/S23 | | 3 & 5 | 0.0313 | 1.4142 |
| 6 | B38/S238 | | | 0.0413 | 2.0000 |
| 7 | B0123478/S0234678 | B37/S23 | 7 & 9 | 0.0553 | 1.4142 |
| 8 | B378/S23 | | 8 & 10 | 0.0625 | 2.0000 |
| 9 | B37/S23 | | 7 & 9 | 0.0664 | 1.4142 |
| 10 | B0123478/S234678 | B378/S23 | 8 & 10 | 0.0678 | 2.0000 |
| 11 | B37/S238 | | | 0.0838 | 2.0000 |
| 12 | B378/S238 | | | 0.0890 | 2.4495 |
| 13 | B3/S237 | | 13 & 15 | 0.0925 | 1.4142 |
| 14 | B3/S2378 | | | 0.0937 | 2.0000 |
| 15 | B023478/S01234678 | B3/S237 | 13 & 15 | 0.0949 | 1.4142 |
| 16 | B48/S23678 | | | 0.0963 | 3.4641 |
| 17 | B468/S236 | | | 0.0970 | 3.1623 |
| 18 | B03478/S01235678 | B4/S2367 | 18 & 34 | 0.0972 | 2.8284 |
| 19 | B478/S238 | | | 0.0976 | 3.1623 |
| 20 | B01468/S034678 | B367/S1356 | 20 & 136 | 0.0979 | 3.4641 |

*Note*. Duplicate rules occur when a rule must be changed to avoid infinity.

2021): (1) A rule containing B0 and S8 is black/white reversed to avoid infinity. (2) A rule containing B0 but not S8 is split into two new rules to avoid strobing. In Table 6, case (1) applies: Some of the rules are modified to prevent infinity. This modification gives rise to duplicate rules, which give us an opportunity to see how much noise there is in our probability estimates. If the probability estimates in the real-valued vectors were perfectly accurate, then the duplicate rules would be adjacent to each other in the table. The duplicates are close together at the top of the list, but they diverge as we go down the list. We will discuss this divergence after we consider another example.



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

Table 7 shows the nearest neighbors of the rule B03/S23 in the 72-dimensional probability space. In Table 7, case (2) applies: All of the rules are modified to prevent strobing. Unlike in Table 6, there are no duplicate rules in Table 7.

In Table 6 and Table 7, the two final columns show the distances from the target rule in real space and Boolean space. The ranking in column 1 is based on real space alone; the Boolean space is only included for comparison. There are many ties in the Boolean column, which shows that the real space is able to make finer distinctions than the Boolean space. We also see that the Boolean distances yield different rankings from the real-valued distances.

Table 7. This table shows the twenty nearest neighbors of the target rule B03/S23 in the 72-dimensional probability space.

| | | Anti-strobing rule change | | Distance from B03/S23 | |
|---|---|---|---|---|---|
| Rank | Rule | Even rule | Odd rule | Real | Boolean |
| 1 | B03/S23—target rule | B1245678/S0145678 | B56/S58 | 0.0000 | 0.0000 |
| 2 | B038/S23 | B124567/S0145678 | B56/S058 | 0.0215 | 1.4142 |
| 3 | B037/S23 | B124568/S0145678 | B56/S158 | 0.0237 | 1.4142 |
| 4 | B0378/S23 | B12456/S0145678 | B56/S0158 | 0.0317 | 2.0000 |
| 5 | B037/S023 | B124568/S145678 | B568/S158 | 0.0365 | 2.0000 |
| 6 | B0378/S023 | B12456/S145678 | B568/S0158 | 0.0452 | 2.4495 |
| 7 | B03/S023 | B1245678/S145678 | B568/S58 | 0.0468 | 1.4142 |
| 8 | B038/S023 | B124567/S145678 | B568/S058 | 0.0525 | 2.0000 |
| 9 | B036/S23 | B124578/S0145678 | B56/S258 | 0.0795 | 1.4142 |
| 10 | B0368/S23 | B12457/S0145678 | B56/S0258 | 0.0816 | 2.0000 |
| 11 | B03678/S23 | B1245/S0145678 | B56/S01258 | 0.0889 | 2.4495 |
| 12 | B0367/S23 | B12458/S0145678 | B56/S1258 | 0.0890 | 2.0000 |
| 13 | B038/S236 | B124567/S014578 | B256/S058 | 0.1020 | 2.0000 |
| 14 | B036/S023 | B124578/S145678 | B568/S258 | 0.1025 | 2.0000 |
| 15 | B03/S236 | B1245678/S014578 | B256/S58 | 0.1042 | 1.4142 |
| 16 | B0368/S023 | B12457/S145678 | B568/S0258 | 0.1045 | 2.4495 |
| 17 | B0378/S236 | B12456/S014578 | B256/S0158 | 0.1094 | 2.4495 |
| 18 | B03/S0236 | B1245678/S14578 | B2568/S58 | 0.1103 | 2.0000 |
| 19 | B03678/S023 | B1245/S145678 | B568/S01258 | 0.1120 | 2.8284 |
| 20 | B037/S236 | B124568/S014578 | B256/S158 | 0.1125 | 2.0000 |

Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

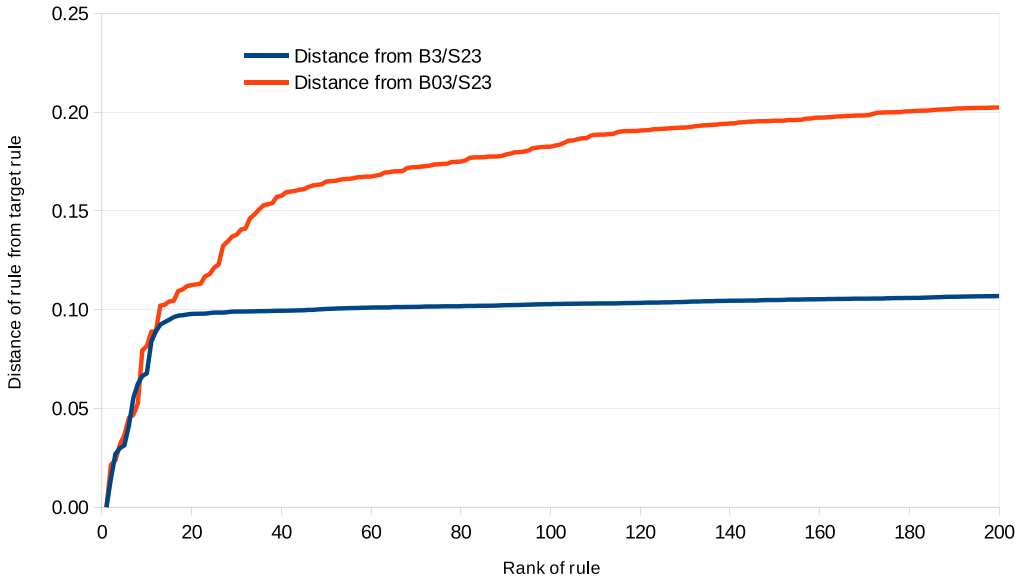


Figure 1. Distance from target rule as a function of rank. This graph shows how the distance of a given rule from a target rule varies as a function of the rank of the given rule.

Figure 1 shows the relation between rank and distance for the 200 most highly ranked rules, with one curve for the target rule B3/S23 and a second curve for the target rule B03/S23. The curve for B3/S23 flattens out at about rank 15, and we can see this flattening also in Table 6, in the column of real distances. This flattening explains why the duplicate rules in Table 6 are near each other until we pass rank 15, when the duplicate rules become distant from each other. When the curve is flat, a small amount of noise in the estimated probability can cause a large amount of variation in the estimated rank. However, the curve for B03/S23 does not flatten out as quickly as the curve for B3/S23. Figure 1 suggests that the rankings for B03/S23 will be reliable at least up to rank 40.

A potential problem with strobing rules is that it seems it should not matter whether the simulation begins at time $t = 0$ or $t = 1$, but changing the starting time will have the effect of swapping the order of the two 36-dimensional parts of the 72-dimensional vector. However, this problem is not likely to arise, since there is no reason to vary the starting time.

## 5 Conclusion

This article offers a new method for examining the behaviors of the family of semi-totalistic cellular automata and for investigating the relations among the members of the family. Given that there are 262,144 semi-totalistic rules, researchers in the field of cellular automata have a large space to explore. In the spirit of Packard and Wolfram (1985) and Eppstein (2010), we hope that our method for measuring behavioral similarity of cellular automata will give researchers useful new ways of viewing, organizing, and understanding the semi-totalistic family.

The general idea of this paper is that the behavior of a deterministic cellular automaton can be captured to some extent by a real-valued probability vector, which expresses the likelihood of various state transitions, given a random soup as a starting point. This idea should be applicable to many other classes of cellular automata, including 1D, 2D, 3D, and higher-dimensional automata, and automata with more than two states.

## Acknowledgments

Thanks to the reviewers of *Artificial Life* for their very helpful advice.



Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023

## References

Bak, P., Chen, K., & Creutz, M. (1989). Self-organized criticality in the 'Game of Life'. *Nature*, *342*(6251), 780–782. https://doi.org/10.1038/342780a0

Eppstein, D. (2010). Growth and decay in life-like cellular automata. In A. Adamatzky (Ed.), *Game of Life Cellular Automata* (pp. 71–97). Springer. https://doi.org/10.1007/978-1-84996-217-9_6

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life." *Scientific American*, *223*(4), 120–123. https://doi.org/10.1038/scientificamerican1070-120

Golly. (2021, March 22). QuickLife: B0 emulation. SourceForge: https://golly.sourceforge.net/Help/Algorithms/QuickLife.html#b0emulation

LifeWiki. (2021a, March 22). *Catagolue*. LifeWiki. https://www.conwaylife.com/wiki/Catagolue

LifeWiki. (2021b, March 22). *Universal constructor*. https://www.conwaylife.com/wiki/Universal_constructor

Packard, N. H., & Wolfram, S. (1985). Two-dimensional cellular automata. *Journal of Statistical Physics*, *38*(5/6), 901–946. https://doi.org/10.1007/BF01010423

Rendell, P. (2016). *Turing machine universality of the Game of Life*. Springer. https://doi.org/10.1007/978-3-319-19842-2

Trevorrow, A., Rokicki, T., Hutton, T., Rowett, C., Hutton, T., Greene, D., Summers, J., Verver, M., Munafo, R., Bostick, B., & Lee, D. (2021, March 22). Golly. SourceForge: https://golly.sourceforge.net/

Turney, P. D. (2021, March 22). Similarity of cellular automata: Source code, GitHub, https://github.com/pdturney/similarity-of-cellular-automata

Downloaded from http://direct.mit.edu/artl/article-pdf/27/1/62/2020412/artl_a_00337.pdf by guest on 08 September 2023