Original Research Article

# Toward Automated Identification of Turing-Complete Cellular Automata via Structural and Complexity Analysis

Alexander L. Malchev

*Sofia High School of Mathematics, Iskar St. 61, Sofia, 1000, Bulgaria*

## ABSTRACT

This research presents a unified, quantitative framework for analyzing the complexity and computational potential of Cellular Automata (CA). By combining the Complexity Score (CS) with the Behavioral Index (BI), the method systematically identifies rules that generate both complex structures and non-trivial interactions - features that recur across known constructions of Turing-complete systems and motivate their use as practical search criteria. This methodology provides a reproducible approach to estimating unbiased simulation parameters, ensuring robust measurements regardless of grid size, time step, or random seed. This reduces variability and enhances comparability across different CA rules and dimensions. The framework applies to all N-dimensional CAs, highlighting edge-of-chaos characteristics. Behavioral analysis allows detailed tracking of object dynamics and collision interactions, providing insights that are not captured by structural metrics alone. Overall, this approach lays a foundation for scalable, quantitative exploration of CA rule spaces and for the systematic identification of candidate universal automata, enabling efficient studies in this area and advancing practical uses of CAs.

**Keywords:** Cellular Automata; Turing-completeness; Complexity; Behavioral Analysis; Object Clustering; Parameter Determination

## INTRODUCTION

A Cellular Automaton (CA) is an N-dimensional grid of cells with a fixed number of states, a system in which cells evolve by interacting with their neighbors according to a predefined rule (1, 2). One-dimensional two-state CAs are called Elementary Cellular Automata (ECA). Figure 1 shows examples of two-state CA grids.

The essence of such systems is that simple local interactions can emulate complex real-life processes. For example, certain 3D CA rules have been shown to model tumor growth with surprising accuracy (3). Research in this field is vital for understanding complex systems that are difficult to analyze with traditional mathematical or scientific tools, while also enabling predictions of how they behave under different initial conditions (4).

Not all cellular automata, however, exhibit equally interesting behavior. Some rules produce trivial dynamics, such as configurations in which every cell converges to a uniform state of all zeros or all ones (1). While valid, such systems offer little insight into the study of complexity or real-world modeling (4, 5). By contrast, other rules generate persistent structures, chaotic behavior, or long transients, making them valuable for both scientific applications and computational theory (1, 2). The distinction between trivial and complex dynamics underscores the need for

**Corresponding author:** Alexander L. Malchev, E-mail: alex.malchev@abv.bg.



**Accepted** January 2, 2026
https://doi.org/10.70251/HYJR2348.41199216

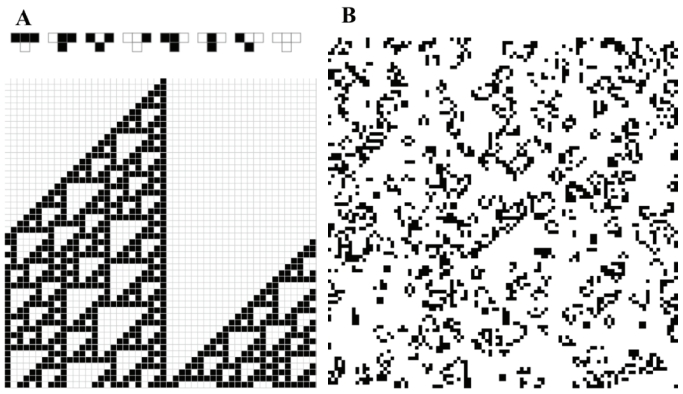


***Figure 1.*** ***Examples of Two-State Cellular Automata. (A)*** *Space-time evolution of Rule 110, an ECA, in which each cell has two possible states and updates according to a local rule indexed by a binary encoding of its neighborhood transitions (shown above the time-space diagram). ECAs are numbered from 0 to 255 based on this encoding.* ***(B)*** *Evolution of Conway's Game of Life, a two-dimensional, two-state cellular automaton in which each cell updates based on the number of active neighbors in its Moore neighborhood (the eight surrounding cells within a radius of one lattice unit). In both panels, black and white cells represent the two possible cell states.*

a reliable measure of complexity. Inadequate or poorly calibrated complexity measures may lead to inconsistent classification of CA rules, misidentification of genuinely interesting dynamics, and inefficient exploration of large rule spaces. As a result, robust and reproducible methods for evaluating complexity are essential for meaningful comparisons across studies.

Complexity in cellular automata is not a straightforward quantity to extract. It indicates whether a system exhibits chaotic, unpredictable behavior or collapses into overly simple dynamics with little interaction between input and output, and it cannot be calculated directly. One of the most widely recognized classification schemes is Wolfram's four classes of complexity, which categorize CA rules as fixed, periodic, chaotic, and complex (1). While useful, this scheme is qualitative and relies on subjective interpretation. Quantitative approaches typically rely on a set of metrics, including Shannon Entropy (SE), compressibility (Compression Ratio or CR), Lempel–Ziv complexity (this paper refers to the Lempel-Ziv algorithm from 1976, or LZ76), and Lyapunov exponents (6, 7, 8, 9). Each of these measures captures essential aspects of CA behavior, but their results are often sensitive to experimental parameters such as grid size, number of timesteps, and initial conditions (10). As a result, comparability across studies remains limited. Most existing studies evaluate complexity using individual metrics or small, uncalibrated combinations of measures, with experimental parameters chosen in a specific manner (1, 7, 9, 10). This practice makes reported complexity values strongly dependent on implementation details rather than intrinsic system dynamics, hindering reproducibility and cross-study comparison. Consequently, there is a lack of a unified, systematically calibrated framework that integrates multiple complementary complexity measures into a consistent quantitative assessment.

In this research, a composite framework for measuring complexity, called the Complexity Score (CS), is proposed. It integrates multiple complementary metrics and introduces systematic calibration of experimental parameters. By reducing sensitivity to arbitrary choices such as grid size, simulation length, and the number of seeds, the framework improves reproducibility and comparability across studies and enables the integration of individual complexity measures, accounting for their differing sensitivities and potential disagreements when applied to the same system. The novelty of this contribution lies not in proposing new complexity metrics but in their structured integration and normalization within a unified evaluation pipeline. The framework is general and applicable to N-dimensional automata, but this work focuses on one-dimensional cellular automata as a test case.

Complexity alone, however, does not guarantee computational universality. Cellular automaton may exhibit high entropy or long transients yet still lack the capacity to support structured information processing (1, 9, 11). For computational universality to emerge, complex dynamics are typically organized to reliably create, transmit, and transform information over time (2, 12). In cellular automata, this organization is evident in higher-order behavioral phenomena, particularly the creation, persistence, and interaction of localized structures (2, 10, 12).

To capture this higher-order dynamic, the paper introduces a Behavioral Analysis framework that quantifies the motion, morphology, and structural stability of these localized structures (often referred to as "objects" or "gliders" (10)), as well as their interactions and collisions. Although such structures have been extensively studied under specific rules using qualitative or manual analysis, no general quantitative method exists for detecting them and assessing their prevalence and dynamical richness across large rule spaces (2, 12).

By formalizing behavioral features into measurable quantities, the proposed Behavioral Index (BI) enables systematic, scalable comparison of emergent structures (10). By combining the CS with the BI, this research establishes a reproducible quantitative framework for identifying cellular automata with potential Turing-completeness, based on the known characteristics of established computational rules (2, 12).

The long-term motivation for this approach is to use the combined complexity and behavior scores as a filter to identify candidate rules capable of universal computation. The space of possible cellular automata grows combinatorially with dimensionality, neighborhood size, and the number of states, making exhaustive manual exploration infeasible (1, 13). Historically, certain cellular automata, such as Conway's Game of Life (CGoL) and Rule 110, have been proven to be Turing-complete, but these are rare discoveries made through decades of manual exploration (2, 12). A scalable, quantitative evaluation of both complexity and behavioral richness narrows the vast space of possible rules and highlights those with the greatest potential for universality.

## LITERATURE REVIEW

Over the years, many approaches have been proposed to quantify complexity in cellular automata. Widely used measures include Shannon Entropy, which quantifies information content in space–time patterns; compression-based measures, which estimate structural regularity by analyzing the compressibility of CA evolutions; Lyapunov exponents, which quantify sensitivity to perturbations; and Damage Spread Rate (DSR), which assesses a rule's stability with respect to its initial input (6, 7, 9, 11). Each metric highlights a different aspect of complexity, but none fully captures it. Results are often sensitive to experimental conditions, leading to inconsistent comparisons across studies (7).

Beyond complexity itself, a fundamental question is whether a CA is Turing-complete. Unlike entropy or compressibility, there is no general algorithm for inferring computational universality, nor is there a universal test for determining it (14). Nevertheless, a small number of cellular automata have been proven universal. In one dimension, the best-known example is Rule 110, which has been shown to emulate a cyclic tag system (2). In two dimensions, the most famous case is Conway's Game of Life, where gliders and other persistent structures can be arranged to simulate arbitrary logic circuits (12). These discoveries were made through painstaking analysis and manual experimentation, underscoring the need for scalable, quantitative methods to identify candidate universal automata.

More recent work has begun to focus not only on raw complexity but also on behavioral richness - the diversity and interaction patterns of emergent objects (10). This perspective, adopted here, views computation in CAs as emerging from structural dynamics: the ways localized entities move, collide, and transform. Quantifying these processes through Behavioral Analysis complements traditional complexity metrics by offering a more direct measure of computational potential.

Another persistent challenge is the stochastic dependence of measured results on initial conditions. Because most CA experiments use randomized initial states, two runs of the same rule may yield divergent outcomes (7, 9, 10, 11). Averaging over many seeds mitigates this effect but requires significant computational effort. This framework formalizes the process by introducing convergence criteria for grid size, time steps, and the number of seeds, ensuring that all complexity and behavioral measures are statistically robust and reproducible.

## METHODS AND MATERIALS

This research employs a three-phase analytical framework to identify candidate Turing-complete Cellular Automata rules systematically. This framework progresses from establishing a robust simulation environment to defining a holistic complexity measure, and finally, to performing a detailed behavioral analysis of object interactions, a potential signature of universal computation (2, 10, 12, 15).

### Establishing robust simulation parameters

The foundation of reproducible complexity analysis rests on simulating CA rules with statistically converged parameter values. Without a systematic determination of the simulation scope, quantitative metrics risk capturing transient behaviors or statistical anomalies rather than the stable, intrinsic properties of the CA rule itself. To address this, the minimum values of several parameters that affect CA simulations are considered. They include: spatial extent (S), the size of the CA grid; evolutionary time (T), the number of evolutionary steps that form a space-time diagram of the history of cells updated during the simulation; and number of seeds (R), the number of trials that are run for each rule to lower noise across

metrics. These optimized values ($S^*$, $T^*$, $R^*$) mitigate initialization bias and ensure that ensemble averaging yields generalizable results.

A pilot parameter-determination procedure comprising three sequential stages - Size Testing, Step Testing, and Seed Estimation - is used to identify converged values. Convergence for size and step count is assessed using a relative tolerance threshold $r_{tol} = 0.08$ for selected metrics (Shannon Entropy, Compression Ratio), while the number of seeds is chosen to satisfy a predefined confidence interval $E = 0.05$. The full algorithmic details, including the computation of relative fluctuations for all metrics, the determination of the minimum ensemble size, and the calculation of the uncertainty and confidence for each metric, are provided in Appendix A.

**Composite Complexity Score**

With the simulation environment standardized, the next goal is to derive a Complexity Score that objectively measures the system's dynamic richness. This quantitative index serves as a crucial initial filter, enabling comparisons across various studies and providing an objective basis for rule classification (7). The CS is constructed by integrating four complementary metrics, computed using the globally defined parameters.

CS utilizes the following metrics:

Shannon Entropy

Shannon Entropy captures the aliveness and information content of the CA's space-time diagram (6). Given a space-time array $X$ of cell state $s \in \{0,1,....,N_s-1\}$, the metric $S_{SE}$ is defined as the normalized output of the Shannon Entropy $S_{SE} \in [0,1]$, by dividing by $log_2 N_s$:

$$S_{SE}(X) = -\frac{1}{log_2 N_s} \sum_{s=0}^{N_s-1} p(s) log_2 p(s)$$

Compression Ratio

The Compression Ratio identifies structural regularity by measuring the compressibility of the system's evolution (7). Let $|C(X)|$ be the compressed length and $|X|$ the raw length of the serialized configuration $X$. The Compression Ratio is:

$$S_{CR}(X) = \frac{|C(X)|}{|X|}$$

For this paper, the compression algorithm of choice is LZMA; however, other algorithms could also be practical (7, 16).

Lempel-Ziv Complexity (LZ76)

Lempel–Ziv complexity approximates the algorithmic information content by counting the number of distinct substrings encountered during the parsing of the serialized CA configuration $L$ of length $n$:

$$S_{LZ76}(L) = \frac{c(L)}{n/log_2 n}$$

Where $c(L)$ is the phrase count (8, 17). Dividing by $n/log_2 n$ normalizes $S_{LZ76} \in [0,1]$.

Damage Spread Rate

The Damage Spread Rate quantifies the system's dynamical sensitivity to initial perturbations – a proxy for chaotic instability (11). It measures the mean fraction of divergence ($\delta_t$) over $T$ time steps between two nearly identical CA instances $A$ and $B$:

$$S_{DSR} = \frac{1}{T} \sum_{t=1}^{T} \delta_t, where\ \delta_t = \frac{1}{N} \sum_{i=1}^{N} [A_i(t) \neq B_i(t)]$$

Normalization and edge-of-chaos filtering

Each metric $S_i$ is first normalized $[0,1]$ using robust percentile scaling to stabilize scaling and mitigate outliers:

$$S' = \frac{S - P_5(S)}{P_{95}(S) - P_5(S)}$$

Additionally, bell-shaped filters are applied to $S_{SE'}$ and $S_{DSR'}$ to weight the results towards the "edge of chaos" ($\mu = 0.5$, $\sigma = 0.2$), the intermediate regime where complex computation typically emerges (1, 7, 9, 10):

$$B(S) = e^{-\frac{(S'-\mu)^2}{2\sigma^2}}$$

The Complexity Score formula

The final Complexity Score is designed to favor systems exhibiting both structural complexity ($S_{CR'}$ and $S_{LZ76'}$) and high activity ($S_{SE'}$ and $S_{DSR'}$).

First, a structure measurement $S_{str}$ is defined as the geometric mean, as both metrics complement each other:

$$S_{str} = \sqrt{S_{CR'} \cdot S_{LZ76'}}$$

The CS is then the square root of the product between $S_{SE'}$, $S_{str}$ and $S_{DSR'}$:

$$CS = \sqrt{S_{SE'} \cdot S_{str} \cdot S_{DSR'}}$$

Error propagation for CS

The method for estimating the error propagation is shown in Appendix B. The final formula for $\Delta CS$ is:

$$\Delta CS = CS \cdot \sqrt{\left(\frac{1}{4}\frac{\Delta S_{CR'}}{S_{CR'}}\right)^2 + \left(\frac{1}{4}\frac{\Delta S_{LZ76'}}{S_{LZ76'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{SE'}}{S_{SE'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{DSR'}}{S_{DSR'}}\right)^2}$$

With all $\Delta S_i$ being the error defined in Appendix A. This method is then run for each rule, and the error variation can depend either on the input, the stability of the rule itself, or both.

## Behavioral analysis to determine the Behavioral Index

Defining BI

Turing-complete CAs are not solely dependent on complexity, and the correlation between the two is not sufficient to assess computational capacity. Computation in CAs can arise from object formation and collision interactions – localized structures that propagate, interact, and sometimes encode logical behavior (10, 12). To capture this, a Behavioral Index is introduced, built upon detailed tracking and analysis of emergent objects.

The first step of clustering objects is detecting those structures in the initial input. Then, there needs to be a way to track the objects through the steps, but running the same algorithm at each step leads to slow detection and also inaccurate results. Because objects are moving, there is no definitive way to identify where the object from the previous step went. If the centers of mass for each object are used to estimate the new position, there would be even more inaccuracies due to the shapes constantly changing; thus, the centers would always move around, and when several objects are close to each other, they can be swapped by mistake (10). An alternative approach would be to run the clustering algorithm only at the beginning and modify the CA, so it still has a rule table for two states, but with a grid of N states (N>2). Each object is assigned a unique state (ID) to help track it. The CA still functions as a 2-state CA, but the new cells are determined by:

$S_{new} = S_{old} \cdot S_{ruletable}$, where $S_{ruletable}$ can be either 0 or 1

Figure 2 showcases this new dynamic with two distinct objects (both alive and treated as such):

***Figure 2. Example of Clustered Objects with Different States.*** *The above objects are gliders from the Conway Game of Life. A clustering algorithm was run, and the objects were assigned new states in the copy of the CA grid. Their states also act as IDs for the objects, represented by different colors, which allow robust tracking for behavioral analysis.*

This method of storing IDs for each object makes it easy to detect two types of collisions: *split* and *merge*. *Merges* can be detected when there is a neighborhood in which there are more than two unique IDs. As the old ID interacts with other IDs, it is assigned as -1. When objects collide, they are considered dead to avoid future conflicts for cells. Their cells are given to newborn objects with their new corresponding ID.

Another collision type that can be detected is a *split*. They are located by running the cluster algorithm on the new N-state CA and identifying several clusters with the same ID. This indicates that in the previous step, the object with the same ID collided with itself, so it can be removed, and new object IDs can be assigned (again to avoid conflicts in cells and confusion in object movement). Figure 3A shows a future collision of two gliders, and Figure 3B shows the aftermath with new objects born from the merge:

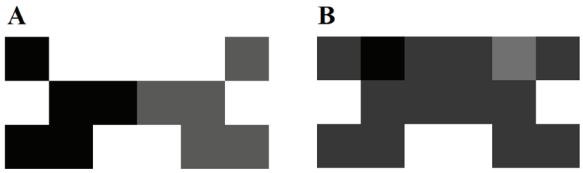


***Figure 3. Example of a Collision of Two Gliders in Conway's Game of Life.*** *Colors denote distinct cell states.* ***(A)*** *Two distinct gliders were identified by the clustering algorithm prior to interaction, each assigned a unique state serving as an object identifier.* ***(B)*** *The configuration after the collision, where the original gliders are removed, and new objects are generated as a result of the interaction. This figure illustrates how object collisions are detected and resolved, enabling quantitative analysis of interaction outcomes used in the Behavioral Index.*

This way, an efficient method is defined for storing information about the objects involved in collisions and their outcomes.

Each tracked object $o$ is characterized by a nine-dimensional feature vector $f_o$, categorized into three main groups representing Kinematics (K), Morphology (M), and Structure (S). These categories quantify complementary aspects of object behavior:

$$f_o = [\underline{v}, \sigma_v, DC, \underline{A}', \underline{P}', T_{period}, H_{struct}, Var, Pers]$$

$\underline{v}$: mean speed of the object.
$\sigma_v$: standard deviation of speed (stability of motion).
$DC$: directional consistency (persistence of travel direction).
$\underline{A}'$: area growth rate (mean normalized change in area).
$\underline{P}'$: perimeter growth rate (mean normalized change in perimeter).
$T_{period}$: periodicity of internal oscillations.
$H_{struct}$: entropy of structural states over time.
$Var$: variability of internal structure.
$Pers$: persistence of form (temporal stability).
Each group's score is aggregated within its category:

$K = mean\ (\underline{v},\ \sigma_v,\ DC)$,
$M = mean\ (\underline{A}',\ \underline{P}')$,
$S = mean\ (T_{period},\ H_{struct},\ Var,\ Pers)$,

Objects are clustered using the DBSCAN algorithm, based on Euclidean distance in normalized feature space, grouping similar behaviors into cluster prototypes $C_k$ (18).

The Object Behavioral Complexity Score (OBCS) is defined as:

$$OBCS = \prod_k \left(\frac{K + M + S}{3}\right)^{p_k}$$

Where $p_k = |C_k| / N_{objects}$ is the cluster's relative frequency.

The Collision Behavioral Complexity (CBC) quantifies the diversity and stability of object interactions. It combines diversity of interaction (via type and transition entropy) and stability of interaction (via continuity and balance metrics), capturing the extent to which collisions are heterogeneous, predictable, and balanced. This approach is motivated by studies of glider collisions in Rule 110, where structured interactions implement information transformations (15).

$$CBC = \sqrt{\left(\frac{H_{type} + H_{trans}}{2}\right) \cdot \left(\frac{\underline{Cont} + \underline{B}}{2}\right)}$$

$H_{type}$: Measures the diversity and unpredictability of input–output mappings between object types across all collisions. It is computed as the normalized Shannon Entropy of observed input–output type transitions.

$H_{trans}$: similarity between the input and output object-types in a collision by comparing the normalized feature vectors of their cluster prototypes.

$\underline{Cont}$: similarity between the input and output object-types in a collision by comparing the normalized feature vectors of their cluster prototypes.

$\underline{B}$: symmetricity of a collision in terms of object count, penalizing collisions that create or destroy many objects while rewarding near-balanced transformations.

Finally, the Behavioral Index integrates OBCS and CBC after being normalized across rules (like CS):

$$BI = \sqrt{OBCS' \cdot CBC'}$$

This ensures that a high BI reflects systems capable of generating both *complex, diverse structures* and *non-trivial, rule-dependent interactions*, features that recur in known constructions of universal cellular automata and motivate their use as practical search criteria. In Figure 4, this framework is explained visually through a flowchart:

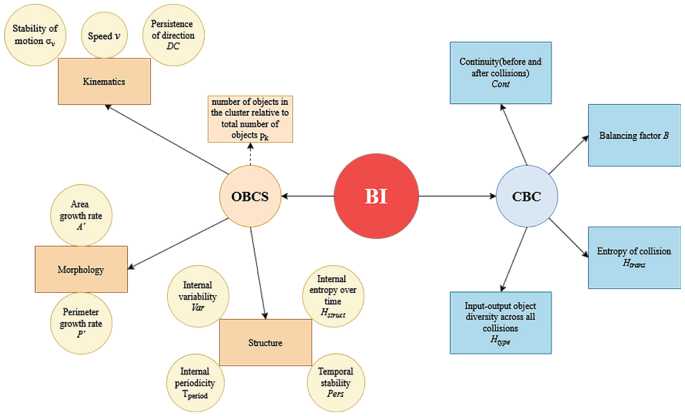


***Figure 4. The BI Framework Explained Visually.*** *The diagram shows the structure of the Behavioral Index framework for evaluating cellular automaton behavior. Metrics related to object-level structural properties contribute to the OBCS and are highlighted in light yellow, while metrics quantifying collision dynamics and interaction outcomes contribute to the CBC and are shown in blue. These components are combined to form the final Behavioral Index, providing a unified quantitative measure of object and collision characteristics for a specific CA.*

Bootstrapping BI values to estimate error

As BI is largely dependent on input, the simplest way to estimate error is by bootstrapping. For each bootstrap sample $i = 1, \ldots, N$, let $OBCS_i$, $CBC_i$ be the values of OBCS and CBC for the given run of the rule. The BI for that sample is calculated as:

$$BI_i = \sqrt{OBCS_i \cdot CBC_i}$$

Then, each BI value is taken to estimate the mean:

$$\underline{BI} = \frac{1}{N}\sum_{i=1}^{N} BI_i$$

And finally, the uncertainty for this metric is the standard deviation for the given rule:

$$\sigma_{BI} = \sqrt{\frac{1}{N-1}\left(BI_i - \underline{BI}\right)^2}$$

This is then run for each rule and calculated based on their performance. Some rules can have larger errors due to a lack of consistent object detection or chaotic patterns.

### Turing-complete probability for CAs

Having established a Complexity Score and a Behavioral Index, the overall probability of a rule being Turing-complete is defined as their geometric mean:

$$P_{TC} = \sqrt{CS \cdot BI}$$

The geometric mean ensures that a high overall score requires both metrics to be elevated. The aim is to prioritize rules that exhibit high complexity and behavioral richness, as these properties are commonly used to narrow the search toward systems that warrant further investigation for computational universality (2, 12).

Error estimation

As $P_{TC}$ is the combination of two independent metrics and by using the same method as for $\Delta CS$, the error $\Delta P_{TC}$ is:

$$\Delta P_{TC} = P_{TC} \cdot \sqrt{\left(\frac{1}{2}\frac{\Delta CS}{CS}\right)^2 + \left(\frac{1}{2}\frac{\sigma_{BI}}{BI}\right)^2}$$

Which is calculated for each rule separately.

## RESULTS

### CA parameter determination

The initial state of the grid is randomized for the purposes of this research, a very common approach to analyzing CAs (6, 7, 10). If a fixed input is chosen, all the measurements and results would be biased towards this specific state of the CA. However, randomization introduces significant noise, and exact measurements are almost impossible to obtain.

As shown in Figures 5A, 5B, 5C, 5D, and 5E, the Complexity Score is highly dependent on the grid size, the number of steps, and the seeds discussed earlier. If

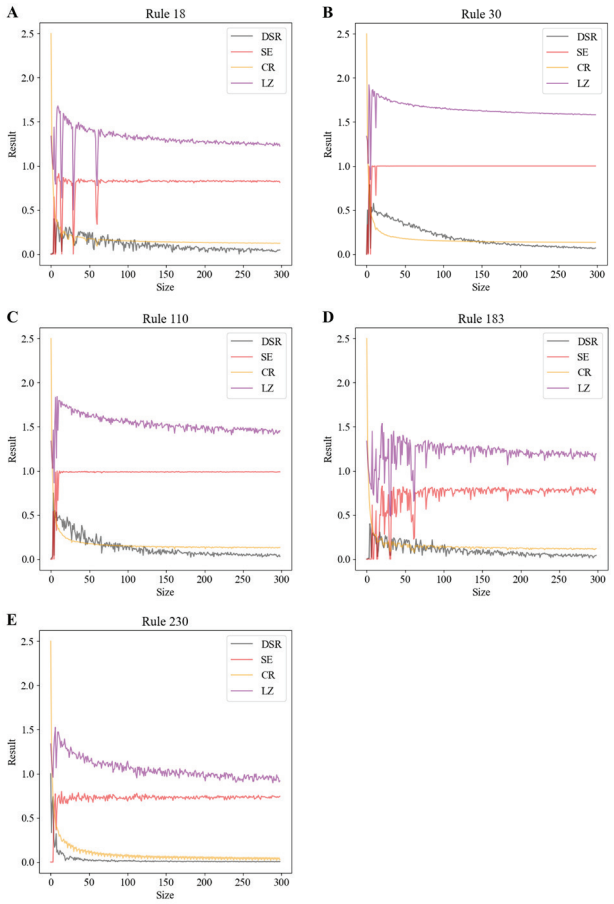


***Figure 5. The Metrics for the Complexity Score Run for a Particular Rule with Respect to the Size of the Grid ($T^* = 50$, $R^* = 3$).*** *DSR is shown in gray, SE in orange, CR in yellow, and LZ76 (on the figure shown only as "LZ") in purple. Panels correspond to different ECA rules:* ***(A)*** *Rule 18,* ***(B)*** *Rule 30,* ***(C)*** *Rule 110,* ***(D)*** *Rule 183,* ***(E)*** *Rule 230. These rules were chosen to demonstrate how the computed metrics vary with the change of grid size variation across a wide range of rules.*

the space-time diagram is too small, the output would be chaotic and undistorted, and DSR would be very high (input-dependent).

And in Figures 6A, 6B, 6C, 6D, and 6E is the same test, but with a changing step count instead.

Although Figures 6C and 6D show some fluctuations in both rules 110 and 183 at the beginning, they are shown to display metrics that are biased towards the small space-time diagrams. Our method for estimating CA parameters aims to lower the flux by adjusting the size and step, not to stimulate input-state changes within the boundaries, and to normalize the collision rate.

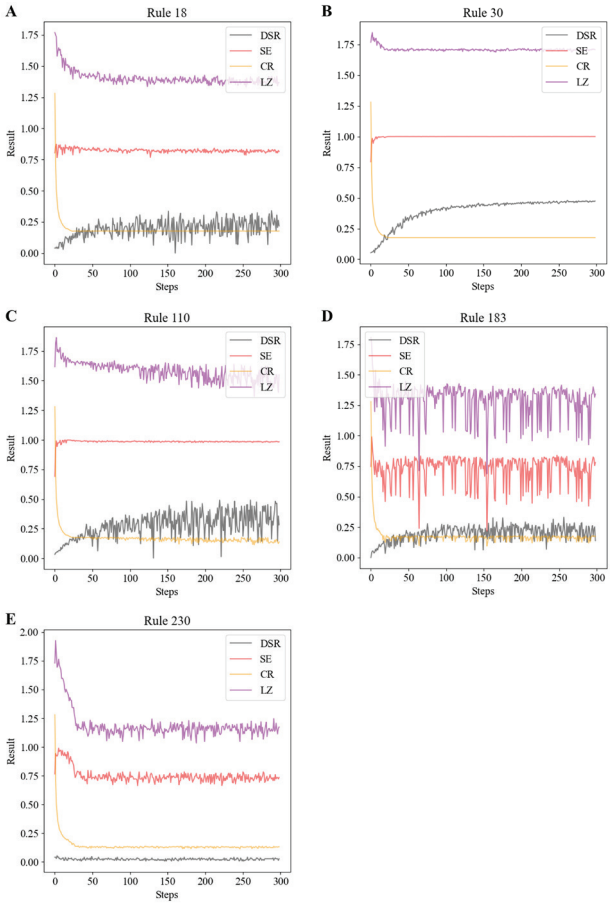


***Figure 6. Correlation Between Metrics of a Rule and Step Count ($S^* = 50$, $R^* = 3$).*** *DSR is shown in gray, SE in orange, CR in yellow, and LZ76 (on the figure shown only as "LZ") in purple. Panels correspond to different ECA rules:* ***(A)*** *Rule 18,* ***(B)*** *Rule 30,* ***(C)*** *Rule 110,* ***(D)*** *Rule 183,* ***(E)*** *Rule 230. Together, these panels illustrate how metric behavior responds to changes in step count across distinct dynamical regimes.*

To reduce fluctuations in rules like *110* and *183*, the number of seeds used to measure CS is adjusted. Figures 7A and 7B show how metrics change with an increase in seed count:

As shown in Figures 7A and 7B, the noise in the readings decreases as the seed count increases, and when all grid parameters exceed certain thresholds, all metrics stabilize, providing the most accurate representation of the CA and enabling analysis of only the rule.

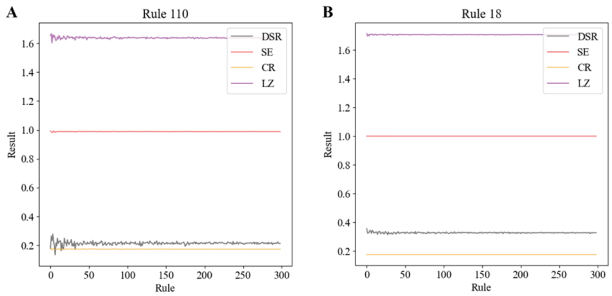


***Figure 7. Normalization of Metrics as Seed Count Increases ($T^* = 50$, $S^* = 3$).*** *DSR is shown in gray, SE in orange, CR in yellow, and LZ76 (on the figure shown only as "LZ") in purple. Metrics computed for each cellular automaton are shown as the number of random initial seeds increases.* ***(A)*** *Rule 110 metrics stabilize, providing an accurate representation of the CA.* ***(B)*** *Noise in Rule 18 metrics also decreases with more seed sampling.*

## Complexity Score

ECA rules

After running the normalization of all metrics and then calculating the CS, the results are shown in Figure 8.

Taking a look at the rule *110* as a proven Turing-complete rule, its CS is around *0.14* (2). The rules that have a higher or equal score are: *18, 107, 110, 137, 146, 182, 183*. And here are the rules around the score *0.14* within $\epsilon = 0.01$: *41, 97, 107, 110, 121, 124, 137, 193*. According to a survey of ECA rule-space classification, many of these rules have been assigned to Wolfram's four classes: rules *18, 137, 146, 182, 183* are all class III; rule 107 is class II, and rule 110 is class IV (2, 13, 19).

Notably, several of these rules lie close to the edge of chaos, where computational universality is most often observed (9). This indicates the Complexity Score may tend to highlight Class IV or Class III–IV boundary rules.

Here are all the normalized metrics for this simulation:

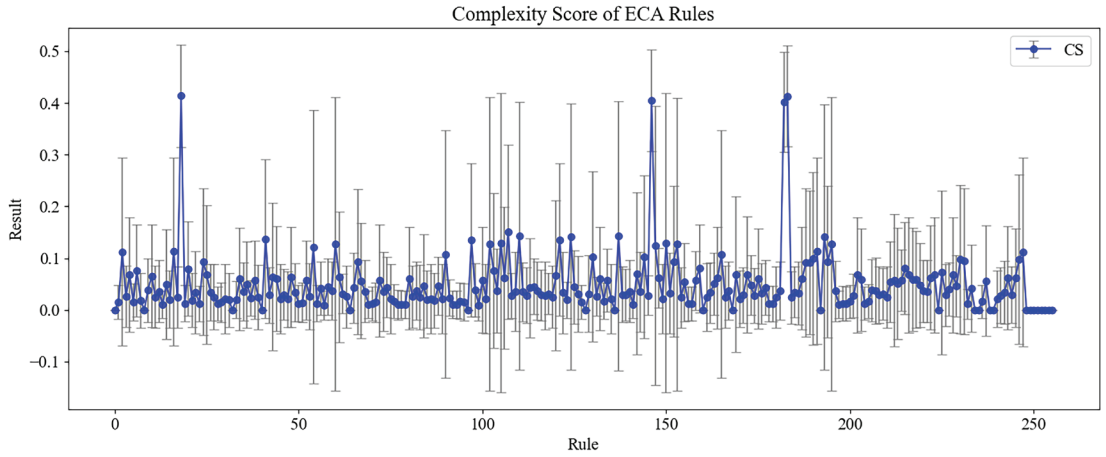


***Figure 8.*** ***Complexity Score for ECA Rules ($S^*$ = 152, $T^*$ = 156, $R^*$ = 65).*** *Error bars are displayed in gray. Each rule's CS quantifies the structural complexity of its space-time patterns. The average score is 0.0472 with median: 0.033956. With CS = 0.0 were 24 rules; CS ≥ 0.1 were 28 rules: 2, 16, 18, 41, 54, 60, 90, 97, 102, 105, 107, 110, 121, 124, 130, 137, 144, 146, 147, 150, 153, 165, 182, 183, 191, 193, 195, 247; and CS ≥ 0.2 there were 4 rules: 18, 146, 182, 183.*

As shown in Figure 9, neighboring rules exhibit fluctuating matrices, suggesting varying complexity across neighbors. This leads to the formation of spikes in certain points of the rule space, which can be seen in Figure 10.

As the rules themselves are not particularly complex, there is an obvious exaggeration on the CS's part. The problem here is the system's fragility from its initial state, regardless of grid size or number of steps. These can be visualized with the high DSR - the higher the DSR value, the more affected the output is by the input.

The Complexity Score can also be used to detect similar rules (13). Examples are: Rule 18 ($CS_{18} = 0.4 \pm 0.09$) and Rule 183 ($CS_{183} = 0.4 \pm 0.09$), Rule 35 ($CS_{35} = 0.03 \pm 0.05$) and Rule 49 ($CS_{49} = 0.03 \pm 0.05$), Rule 110 ($CS_{110} = 0.14 \pm 0.26$) and Rule 124 ($CS_{124} = 0.14 \pm 0.26$).

These equivalence relationships indicate that the Complexity Score could possibly capture functional similarity among behaviorally related rules, suggesting its potential use as a rule-space clustering metric.

This implies that CS may not only be used for potential Turing-completeness, but also as a rule comparator.

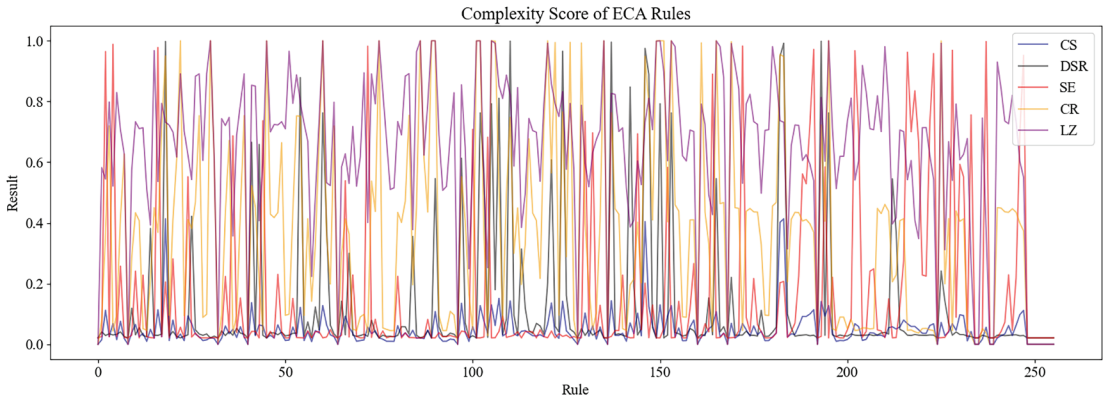


***Figure 9.*** ***Normalized Complexity Metrics and CS ($S^*$ = 152, $T^*$ = 156, $R^*$ = 65).*** CS is shown in blue, DSR in gray, SE in orange, CR in yellow, and LZ76 (on the figure shown only as "LZ") in purple. Each metric is normalized to allow comparison across rules. CS is included for reference. Differences between metrics for neighboring rules are clearly visible, highlighting inconsistencies in how individual metrics capture structural complexity.

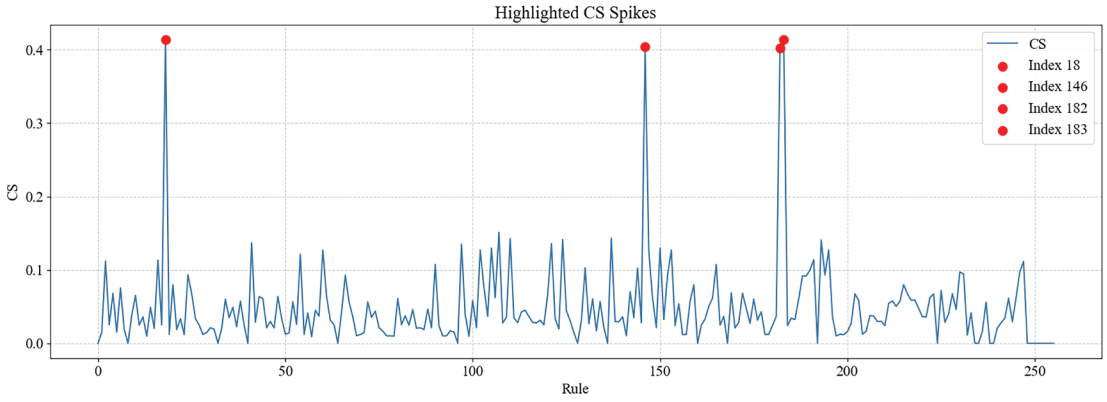


***Figure 10.*** ***Spikes in Simulation.*** *Spikes are highlighted with red dots. The measurement was run several times and independently calculated ($S^*$ = 152, $T^*$ = 156, $R^*$ = 65). All the scores were consistent, and the most visible anomaly is the spikes. Those high CS scores are for the following rules: 18, 146, 182, 183.*

DSR as an error measurement

DSR shows the instability of a system relative to its input (11). This can be considered a complementary metric and an error indicator for the measurements. Figure 11 presents the DSR and CS for each rule, using DSR as the metric rather than a penalty.

As shown in Figure 12, as DSR is normalized, the penalty of a DSR value increases, and almost all of the CAs have a lowered Complexity Score. This provides insufficient data for research and exaggerates the input bias.

## Applying CS to 2D CA rules

Practical limitations

For a $D$-dimensional, $N_s$ state CA with a neighborhood radius $r$, the total number of possible patterns is

$$P = N_s^{r^D}$$

And the total number of rules is

$$N_R = N_s^P$$

Then, for a CA with $D$ = 2, $N_s$ = 2 and Moore's

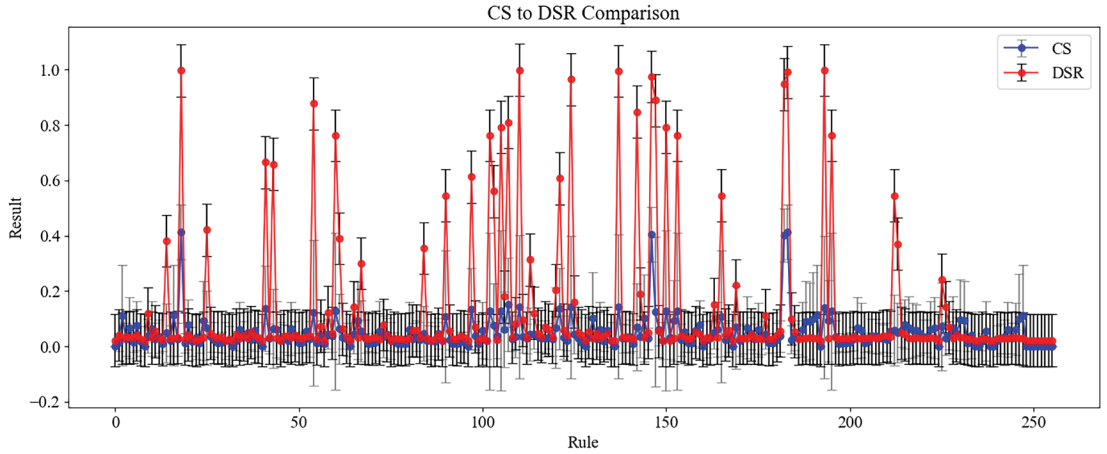


***Figure 11.*** ***CS with DSR (currently adapted).*** *DSR is displayed in orange with black error bars, and CS in blue with gray error bars. DSR here is a metric instead of a penalty. Simulations parameters were $S^*$= 152, $T^*$ =156, $R^*$ = 65. The simulation was run multiple times, and CS values were recalculated accordingly, illustrating how DSR influences the measurement of complexity in selected ECA rules.*

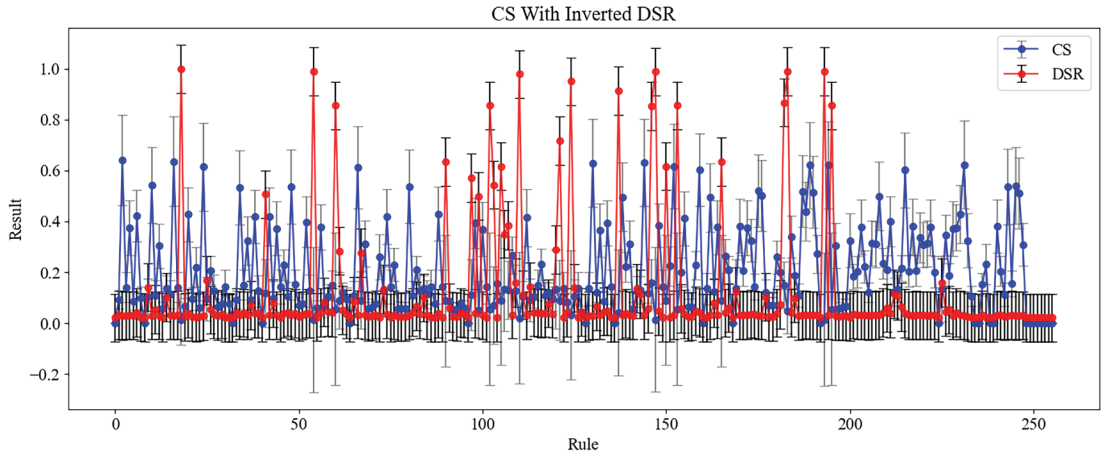


***Figure 12.*** ***CS with Inverted DSR (1-DSR).*** *DSR is displayed in orange with black error bars, and CS in blue with gray error bars. The purpose of this simulation is to show how inverting DSR affects the complexity assessment across selected ECA rules. Normalised DSR leads to a penalty increase and a lowered CS. The simulation was run with* $S^* = 152$, $T^* = 156$, $R^* = 65$.

neighborhood ($r = 3$):

$$P = 2^{3^2} = 512 \Rightarrow N_R = 2^P = 2^{512}$$

With current hardware, it is not possible to analyze all of the 2D rules, and thus, the Complexity Score can only be used for possible correlations between CAs and partially hint at their true complexity, though the error would probably be large.

CGoL and neighbor rules

The results around Conway's Game of Life, shown in Figure 13, suggest that small rule perturbations drastically alter the system's dynamical class. Despite neighboring rules in parameter space often sharing similar entropies, the emergence or absence of self-sustaining objects (gliders, oscillators) is a defining feature distinguishing Wolfram's Class IV dynamics

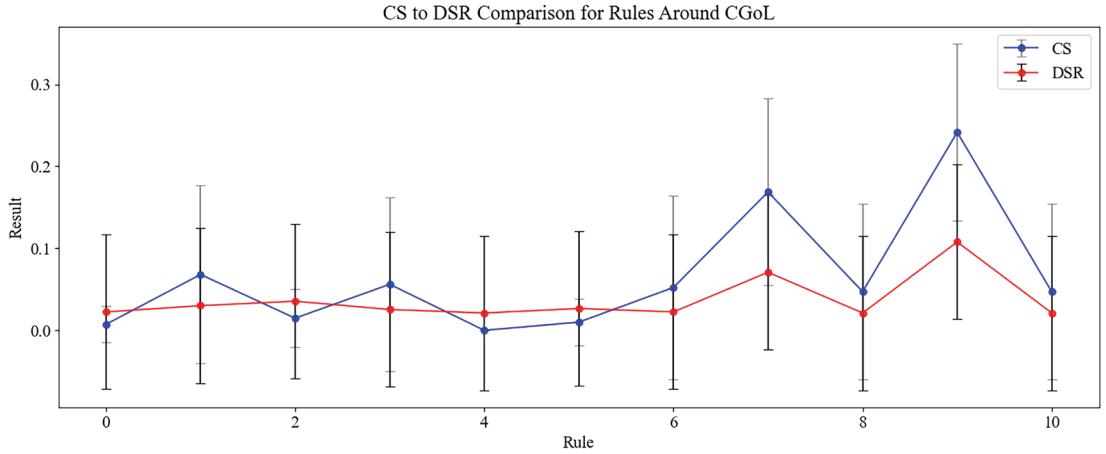


***Figure 13.*** ***Rules Around the Conway Game of Life Rule.*** *CS is shown in blue with gray error bars, while DSR is shown in orange with black error bars. CGoL, corresponding to rule number 6, is compared to neighboring rules based on extracted complexity metrics. Most neighboring rules show higher CS values than CGoL with higher DSR values as well, indicating greater sensitivity to initial conditions relative to CGoL.*

from the more chaotic behavior characteristic of Class III rules (1, 10, 13). This highlights how minor rule-table changes can undermine universality, reinforcing the need to combine the Complexity Score and the Behavioral Index when identifying potentially Turing-complete rules.

**Behavioral Index**

Here is the Behavioral Index for each rule of ECA after OBCS and CBC were normalized:

Among the ECA rules in Figure 14, approximately one-third produced a BI value of *0.0*, signifying negligible behavioral activity. These rules correspond to systems that quickly reach homogeneous or repetitive patterns, which align with Wolfram's Class I and Class II categorizations of cellular automata (3, 13). At the opposite end of the spectrum, a small group of rules demonstrated maximal behavioral complexity. Rules *110, 111, 124*, and *125* each achieved *BI* = 1.0 with errors of around ± *0.03*, while rule *230* reached a BI of approximately *0.99* ± *0.04*. Several additional rules - specifically *60, 61, 62, 63, 102*, and *103* - produced BI values in the range of *0.97* to *0.98*.

This distribution aligns closely with previously documented behavioral analyses of ECA. Rule *110*, in particular, has been rigorously proven to be Turing-complete, and its rich glider and collision structures have been extensively analyzed by de Cómputo, E. S. in 2013 (2, 20). The fact that Rule *110* achieves a *BI* = 1.0 result in this dataset increases the likelihood that the proposed metrics accurately capture both object diversity and collision dynamics characteristic of computational universality. Likewise, Rule 124 - being the mirror reflection of Rule *110* in the ECA rule space - naturally exhibits similar dynamics and correspondingly achieves the same BI value (13).

The slightly lower BI scores observed for rules, such as *60, 61, 62, 63, 102*, and *103* also reflect known properties of these systems. Previous researches describe these rules as exhibiting interacting or transiently complex structures that may eventually settle into simpler configurations (21).

Conversely, the majority of rules with near-zero BI correspond to well-known trivial or periodic behaviors (13). These include rules that quickly reach uniform or repeating configurations, typical of Class I and Class II ECAs.

**Turing-Complete Probabilities for ECA Rules**

Having defined the Turing-complete Probability as the geometric mean of the Complexity Score and the Behavioral Index:

$$P_{TC} = \sqrt{CS' \cdot BI'}$$

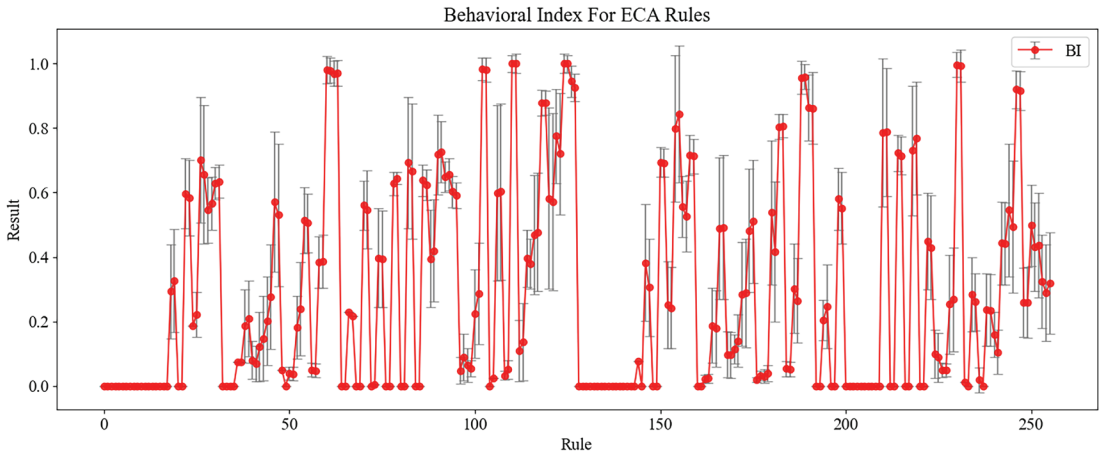


***Figure 14. Behavioral Index for ECA.*** *Error bars are displayed in gray. The BI quantifies the observed dynamical behavior of each rule, with higher values indicating more complex or varied dynamics. The average score is 0.296969 with median: 0.166066. With BI = 0.0 were 84 rules; BI ≥ 0.5 were 78 rules (22, 23, 26, 27, 28, 29, 30, 31, 46, 47, 54, 60, 61, 62, 63, 70, 71, 78, 79, 82, 83, 86, 87, 90, 91, 92, 93, 94, 95, 102, 103, 106, 107, 110, 111, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 150, 151, 154, 155, 156, 157, 158, 159, 166, 180, 181, 182, 183, 188, 189, 190, 191, 198, 199, 210, 211, 214, 215, 218, 219, 230, 231, 244, 245, 246, 247, 251); and BI ≥ 0.9 were rules (60, 61, 62, 63, 102, 103, 110, 111, 124, 125, 126, 127, 188, 189, 230, 231, 246, 247).*

The likelihood of each ECA rule to exhibit computational universality can finally be evaluated. The geometric mean ensures that both components contribute symmetrically, penalizing rules that are high in one metric but low in the other.

The resulting probabilities for all ECA rules are shown in Figure 15.

The results show that the most potentially universal rules are *182* and *183* with both very close probabilities of around *0.57 ± 0.07*. Although they are not proven to be universal, they were consistently ranked among the highest in both CS and BI, suggesting possible computational capabilities. They are followed by rule *110* with a probability of *0.38 ± 0.3* and *146* being *0.4 ± 0.1*. With rule *110* being proven to be universal, the prediction puts it in the top 4, and although the error is high, it consistently was amongst the most highly scoring rules (2). For rule *146*, it hasn't been proven to be universal but still showed consistent readings in both metrics, suggesting possible Turing-completeness.

For nearly identical rules, the scores were similar as well. For rules *60* and *102* the values were both around *0.35* for *110* and *124* the values were around *0.4* for both. However, this consistency amongst mirrored rules is not applied everywhere. The mirror rule *182* is *18*. Its probability of around *0.34 ± 0.09* puts it in eighth place after rules *60, 102, 110, 124, 146, 182*, and *183* all of which are either class III or class IV CAs, and still, its score is *5* rules apart (13).

From Figure 15, the average score is *0.086253* with median: *0.066459*. With $P_{TC} = 0.0$ were 100 rules; $P_{TC} \geq 0.25$ were 22 rules (18, 60, 61, 90, 102, 103, 107, 110, 121, 124, 146, 150, 182, 183, 188, 189, 190, 191, 230, 246, 247); $P_{TC} \geq 0.34$ were 8 rules (18, 60, 102, 110, 124, 146, 182, 183); and were rules (*182, 183*).

## LIMITATIONS

Even though both metrics are evaluated with parameter determination, some degree of input bias inevitably remains. A full exploration of two-dimensional cellular automata is computationally prohibitive: the parameter space is vast, and exhaustively sampling it would require resources far beyond what is feasible for this study. To estimate the methodology's effectiveness, it was primarily applied to ECA rules. They are already well-characterized in the literature, providing a reliable baseline for comparison and a practical means of validating the approach (2, 12, 13).

The nature of Turing-completeness itself creates another limitation. No universal algorithm has been found that proves a CA's computational capabilities for any given rule (2, 12). As a result, the $P_{TC}$ measure cannot offer definitive proof. Instead, it provides a probabilistic indication of the universality of a rule set. A high $P_{TC}$ score suggests that a cellular automaton may have computational capabilities, but it cannot confirm them in a strict formal sense. Thus, while the $P_{TC}$ is a useful heuristic for identifying potentially Turing-complete systems, it should be interpreted as suggestive rather than conclusive.

The last factor is time efficiency. Parameter

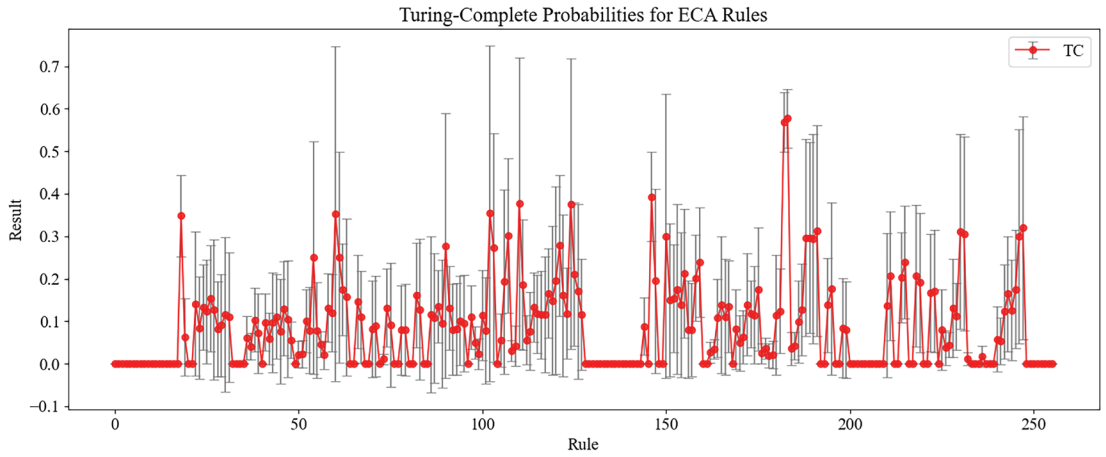


***Figure 15. Turing-Complete Probabilities for ECA Rules.*** *Each value represents the estimated probability that a rule exhibits Turing-complete behavior under the proposed framework. Error bars are shown in gray.*

determination was conducted using limited and relatively coarse initial samples for lattice size and evolution steps due to computational constraints. This resulted in large uncertainties in the estimated Complexity Score. Consequently, the error of the Turing-Complete Probability for Rule 110 is approximately 75%, primarily driven by the high variance associated with its CS value. The large relative uncertainty observed for Rule 110 reflects the tension between finite-sampling statistical metrics and worst-case formal universality proofs. This is not a failure of the proposed framework, but rather an inherent consequence of estimating average-case dynamical richness. Additionally, the use of limited-size and step tests causes the estimation of optimal simulation parameters to become highly input-dependent, leading to substantial fluctuations. While this is not ideal for research purposes, it is resolvable. Future studies employing the Complexity Score should adopt a broader and more systematic exploration of initial lattice sizes and evolution times to obtain more reliable and lower-uncertainty estimates.

## DISCUSSION

### The choice of compression algorithms

For this paper, several algorithms were tested simultaneously. While they all manage to track different patterns, the contrast between them introduces more noise than useful information. Therefore, only one compression algorithm was used - LZMA (7, 16). Using a single, robust compressor ensures that the Complexity Scores remain consistent across different runs and seeds. Moreover, the specific pattern recognition of the algorithm should not be a significant factor, as the primary purpose of this metric is to capture structural repetitiveness, which provides insight into the order of the rule (16).

### Some interesting results

Rule 182 and 183

The characterization of rules 18, 146, 182, and 183 as "anomalies" in Figure 10 refers specifically to their behavior under the Complexity Score alone, where elevated Damage Spread Rate can inflate CS values and produce pronounced spikes. At this stage, these spikes indicate sensitivity to initial conditions rather than validated computational structure, and therefore require further disambiguation. This apparent exaggeration does not imply that all spike rules are false positives, but rather that CS by itself cannot distinguish between pathological instability and productive edge-of-chaos dynamics. The subsequent Behavioral Index analysis resolves this ambiguity: while rule 18 and, to a lesser extent, rule 146 fail to sustain structured object dynamics, rules 182 and 183 retain high scores after behavioral filtering.

The consistently high values obtained for Rules *182* and *183* across all measurements - Complexity Score ($CS \approx 0.40$), Behavioral Index ($BI \approx 0.82$), and the combined probability $P_{TC} \approx 0.57$- suggest that these two rules occupy a particularly interesting region of the ECA rule space. Their scores place them well above typical class III rules in structured complexity, yet also above typical class IV rules in dynamical richness, positioning them near the "edge of chaos" where universal computation is known to emerge (2, 9, 12, 13). Given that neither rule has been formally proven universal, the present framework identifies them as strong candidates for future theoretical investigation.

A striking outcome is the relationship between these high-scoring rules and the behavior of their mirrored counterparts. Rule *183* is the mirror of Rule *18*, and the two display markedly different Turing-complete probabilities: $P_{TC,183} = 0.57$ versus $P_{TC,18} = 0.34$. Despite this numerical gap, Rule *18* still ranks within the upper tier of the distribution. This suggests that certain computational traits persist across reflection. Still, not with complete symmetry - reflection preserves some dynamical signatures but disrupts others, especially those related to directionality and asymmetry of particle interactions.

Even more intriguingly, rule *182* exhibits a similar mirrored relationship with rule 46, producing nearly the same pattern: $P_{TC,182} = 0.57$, $P_{TC,146} = 0.36$. The fact that both high-probability rules (*182* and *183*) have mirrors that fall to almost exactly the same lower value could imply that these rules may share a deeper structural kinship - not just in their raw outputs, but in how the underlying mechanisms for complexity and behavior propagate under reflection. This "paired similarity pattern" indicates that the CS - BI framework is sensitive to subtle features of rule dynamics that are not strictly preserved by syntactic symmetry alone.

Overall, these relationships strongly suggest that rules *182* and *183* exhibit:

1. High structural and dynamical complexity,
2. Stability of those qualities under reflection (though at reduced strength),
3. Distinctive interaction patterns that place them higher than rule 110 in computational potential,

4. And a mirrored-rule signature that might indicate deeper invariants governing universality-adjacent behavior.

This pattern highlights the usefulness of the present method: it does not merely identify isolated high-scoring rules but reveals families of behaviorally related rules whose computational potential may be systematically explored. The repeated appearance of these structural relationships strengthens the case for prioritizing rules *182*, *183* and their mirrored counterparts as promising and underexplored candidates for future universality research.

Rule 110

As the only formally proven universal ECA, rule 110 serves as the central benchmark for evaluating the validity of the proposed metrics (2). Its performance across both the Complexity Score and the Behavioral Index closely aligns with theoretical expectations and provides an important reference point for interpreting the broader results (2).

In the CS metric, rule 110 attains a value of *0.14*, placing it seventh overall and just below several class III rules (13) that score higher due to their increased syntactic and statistical variability. This is consistent with the nature of rule 110: as it belongs to Wolfram's class IV, its structured patterns and interacting gliders should theoretically inflict more regularity than the chaotic rules above it (2, 13). Thus, its moderate CS value captures this subtle interplay of order and disorder, consistent with properties highlighted in constructions of universal computation (2, 12).

In contrast, rule 110 reaches the maximum possible value (*1.0*) in the Behavioral Index. This peak score indicates extremely high dynamical richness, capturing its well-documented ability to generate complex particle interactions, persistent structures, and long-range information transfer (2).

$P_{TC}$ placement or rule 110 relative to its neighbors further strengthens the interpretive reliability of the framework: rules adjacent to 110 - particularly 102, 124, and 146 - also score high in both metrics, mirroring known similarities in their dynamical behavior. This clustering indicates that the CS-BI framework is sensitive to structurally meaningful variations in rule behavior (13). Within this context, the placement of rule 110 below rules 182 and 183 does not contradict its proven universality (2). The $P_{TC}$ metric quantifies computational potential under random initial conditions, favoring systems that naturally generate abundant structures and interactions without engineered inputs. Rule 110's universality, by contrast, relies on carefully constructed initial configurations and sparse, highly constrained particle interactions (20). As a result, its average-case behavioral richness is lower than that of more dynamically prolific rules, even though it is formally universal (2).

Overall, rule 110's performance provides an essential internal validation: the proposed method not only identifies it as a top candidate but also contextualizes its characteristics in a way that aligns with established theoretical knowledge about its computational capabilities (2).

**Turing-complete estimation using Convolutional Neural Network (CNN)**

With the new metrics and Turing-complete estimate, further automating this process adds a third phase that includes artificial intelligence (AI). This is a great candidate for future work for which the groundwork could be laid now.

A CNN could be trained to search for *solutions* to the system. *Solutions* is a summary term for finding algorithmic patterns between frames of the CA. They can be divided into several categories. At a low level, a solution can effectively predict the next cell state. One such solution is already known for this level and is known for each CA - the rule of the CA itself. Mid-level analyzes objects and collisions (tries to create logic gates and variables). This level can be integrated with the behavioral analysis, which has already been established, and only compares, predicts, and creates patterns from collision results. High-level solutions require analyzing the full space-time diagram and predicting correlations between frames a few apart. For example, some CAs, like the rule before CGoL, have a connection between non-consecutive steps. These include other kinds of possible solutions that involve more than just objects and collisions.

The more solutions are found across these levels, the more likely it is for a CA to exhibit computational universality (interpretation of the definition of (14)). However, this does not directly prove Turing-completeness; further formal verification is required, as CNN predictions are emulations rather than exact derivations of the CA's computational capacity.

The issues with this approach are several. The first one is the difficulty of validating a solution (examples of validating solutions (2, 12)); the higher the level, the more difficult it is. There needs to be more research in this part. Secondly, the number of possible solutions is theoretically infinite for each level, as there are no known

boundaries or rules that the solution must obey.

This needs to be researched more thoroughly, as it could provide proof of the Turing-completeness of all CA rules across all dimensions.

## CONCLUSION

This study presents a unified framework for evaluating the complexity and computational potential (14) of Cellular Automata, integrating the Complexity Score with the Behavioral Index. By combining structural metrics with behavioral analysis of emergent objects and their interactions, it is possible to identify rules that support non-trivial dynamics, which can then be used to highlight those that warrant further investigation for potential computational universality.

Our results demonstrate that even minor modifications to rule definitions can dramatically alter a CA's emergent behavior, hinting at the fragility and sensitivity of computational universality. This reinforces the need to consider both structural complexity and the dynamics of interacting entities when assessing computational potential.

The methodology also provides a strong, reproducible approach to standardize simulations, ensuring consistent evaluation across grid sizes, evolution steps, and random initial conditions. Applications to elementary 1D CAs and the Conway Game of Life confirm that the framework reliably depicts rules capable of sustaining persistent, interacting structures from those that produce trivial or chaotic patterns.

Looking forward, this framework opens pathways for efficiently exploring higher-dimensional automata, and for adding AI-driven *solution* recognition to systematically filtered high-probability universality rules in a CA rule space. This work advances a quantitative, reproducible approach for discovering computation in complex discrete systems, providing a foundation for both theoretical study and practical applications of Cellular Automata.

## ACKNOWLEDGEMENTS

I want to thank Polygence and my academic mentor, Arpit Jasapara, for their guidance and support in writing this research paper.

## FUNDING SOURCES

The author declares no funding sources.

## CONFLICT OF INTEREST

The author declares that there are no conflicts of interest related to this work

## REFERENCES

1. Wolfram S. Universality and complexity in cellular automata. *Physica D.* 1984; 10 (1-2): 1-35. https://doi.org/10.1016/0167-2789(84)90245-8
2. Cook M. Universality in elementary cellular automata. *Complex Syst.* 2004; 15: 1-40. https://doi.org/10.25088/ComplexSystems.15.1.1
3. Boondirek A, Triampo W, Nuttavut N. A review of cellular automata models of tumor growth. *Int Math Forum*. 2010; 5 (61): 3023-3029.
4. Santé I, García AM, Miranda D, Crecente R. Cellular automata models for the simulation of real-world urban processes: a review and analysis. *Landsc Urban Plan.* 2010; 96 (2): 108-122. https://doi.org/10.1016/j.landurbplan.2010.03.001
5. Ghosh M, Kumar R, Saha M, Sikdar BK. Cellular automata and its applications. In: Proceedings of the 2018 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS); 2018 Oct; Malaysia. IEEE; 2018; p. 52-56. https://doi.org/10.1109/I2CACIS.2018.8603689
6. Shannon CE. A mathematical theory of communication. *Bell Syst Tech J.* 1948; 27 (3): 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
7. Zenil H. Compression-based investigation of the dynamical properties of cellular automata and other systems. arXiv preprint arXiv:0910.4042. 2009.
8. Lempel A, Ziv J. On the complexity of finite sequences. *IEEE Trans Inf Theory.* 1976; 22 (1): 75-81. https://doi.org/10.1109/TIT.1976.1055501
9. Langton CG. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D.* 1990; 42 (1-3): 12-37. https://doi.org/10.1016/0167-2789(90)90064-V
10. Wuensche A. Classifying cellular automata automatically: finding gliders, filtering, and relating space-time patterns, attractors, and the Z parameter. *Complex Syst*. 1999; 11 (3): 1-28. https://doi.org/10.1002/(SICI)1099-0526(199901/02)4:3<47::AID-CPLX9>3.0.CO;2-V
11. Bagnoli F, Rechtman R, Ruffo S. Damage spreading and Lyapunov exponents in cellular automata. *Phys Lett A.* 1992; 172 (1-2): 34-38. https://doi.org/10.1016/0375-9601(92)90185-O
12. Rendell P. Turing universality of the Game of Life. In: Adamatzky A, editor. Collision-based computing.

London: Springer; 2002; p.513-539. https://doi.org/10.1007/978-1-4471-0129-1_18
13. Li W, Packard N. The structure of the elementary cellular automata rule space. *Complex Syst*. 1990; 4 (3): 281-297.
14. Turing AM. On computable numbers, with an application to the Entscheidungsproblem. *J Math*. 1936; 58: 345-363. https://doi.org/10.1093/oso/9780198250791.003.0005
15. Mora JC, Vergara SV. Rule 110 objects and other collision-based constructions. *J Cell Autom*. 2007; 14: 56.
16. Ziv J, Lempel A. A universal algorithm for sequential data compression. *IEEE Trans Inf Theory*. 1977; 23 (3): 337-343. https://doi.org/10.1109/TIT.1977.1055714
17. Li M, Vitányi P. An introduction to Kolmogorov complexity and its applications. 3rd ed. New York: Springer; 2008. ISBN: 978-0-387-33998-6.
18. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD); 1996 Aug; Portland, OR. p. 226-231.
19. Martinez GJ. A note on elementary cellular automata classification. arXiv preprint arXiv:1306.5577. 2013.
20. de Cómputo ES. On soliton collisions between localizations in complex elementary cellular automata: rules 54 and 110 and beyond. arXiv preprint arXiv: 1301.6258. 2013.
21. Chen F, Shi L, Chen G, Jin W. Chaos and gliders in periodic cellular automaton rule 62. *J Cell Autom*. 2012; 7 (4).

## APPENDIX A. Parameter Convergence

Each rule is first simulated across a range of lattice sizes $S_i \in \{S_1, S_2, S_3, \ldots, S_n\}$. For each size, the mean values of selected observables, specifically Shannon Entropy and Compression Ratio, are recorded. Spatial stability is determined when the relative change between consecutive sizes falls below a tolerance threshold $r_{tol} = 0.08$:

$$\Delta(x_i, x_{i-1}) = \frac{|x_i - x_{i-1}|}{|x_{i-1}| + \varepsilon} < r_{tol}, x \in \{SE, CR\}$$

When two consecutive tests satisfy the condition for all metrics, the smallest stable size $S^* = S_i$ is selected.

Using the previously estimated $S^*$, the automaton is evolved for increasing temporal depths $T_j$, and the same stability criterion is used to identify $T^*$.

Finally, to estimate the required number of random initializations $R^*$, each metric is measured over $R_{pilot}$ random seeds. The minimum number of repetitions needed for a 95% confidence interval with a margin of error $E = 0.05$ is computed as:

$$R^* = \max_i \left\lceil \left( \frac{1.96 \cdot \sigma_i}{E} \right)^2 \right\rceil, \sigma_i = stdev(x_i)$$

Where $i$ iterates over selected metrics {SE, CR, LZ76} and $x_i$ represents the specified metric's value.

Both $r_{tol}$ and $E$ equally determine the error of each metric $\Delta S_i$ where i ∈ {*SE, CR, LZ76, DSR*}:

$$\Delta S_i = \sqrt{r_{tol}^2 + E^2}$$

## APPENDIX B. Error Propagation of the Complexity Score

The CS formula can be rewritten by expressing the nested roots as fractional powers:

$$CS = (S_{CR'} \cdot S_{LZ76'})^{\frac{1}{4}} \cdot (S_{SE'} \cdot S_{DSR'})^{\frac{1}{2}}$$

Taking the natural logarithm gives:

$$lnCS = \frac{1}{4}(lnS_{CR'} + lnS_{LZ76'}) + \frac{1}{2}(lnS_{SE'} + lnS_{DSR'})$$

Differentiating yields the relative change:

$$\frac{\Delta CS}{CS} = \frac{1}{4}\frac{\Delta S_{CR'}}{S_{CR'}} + \frac{1}{4}\frac{\Delta S_{LZ76'}}{S_{LZ76'}} + \frac{1}{2}\frac{\Delta S_{SE'}}{S_{SE'}} + \frac{1}{2}\frac{\Delta S_{DSR'}}{S_{DSR'}}$$

Due to the metrics being independent, the uncertainties are combined in quadrature:

$$\left(\frac{\Delta CS}{CS}\right)^2 = \left(\frac{1}{4}\frac{\Delta S_{CR'}}{S_{CR'}}\right)^2 + \left(\frac{1}{4}\frac{\Delta S_{LZ76'}}{S_{LZ76'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{SE'}}{S_{SE'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{DSR'}}{S_{DSR'}}\right)^2$$

And the final formula for $\Delta CS$ is:

$$\Delta CS = CS \cdot \sqrt{\left(\frac{1}{4}\frac{\Delta S_{CR'}}{S_{CR'}}\right)^2 + \left(\frac{1}{4}\frac{\Delta S_{LZ76'}}{S_{LZ76'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{SE'}}{S_{SE'}}\right)^2 + \left(\frac{1}{2}\frac{\Delta S_{DSR'}}{S_{DSR'}}\right)^2}$$

With all $\Delta S_i$ being the error defined earlier in the methodology section for parameter estimation.