# Graph-Based Representations of Elementary Cellular Automata: A Literature Review

### Abstract

Cellular Automata (CA), particularly Elementary Cellular Automata (ECA), serve as fundamental models for studying complex systems, demonstrating how simple, local rules can generate intricate global behavior. A crucial aspect of analyzing these systems is the method of their representation. This review provides a comprehensive survey of graph-based methodologies used to represent and analyze the 256 ECA rules. We examine two primary, contrasting paradigms: the State Transition Graph (STG), which offers a complete depiction of a system's global dynamics but suffers from exponential complexity, and the De Bruijn Graph (DBG), which provides a compact, fixed-size encoding of the local rule but loses all information about emergent global behavior. We analyze the strengths and inherent limitations of each approach, detailing how their respective graph topologies are used to understand properties such as attractors, reversibility, and surjectivity. Furthermore, we explore modern generalizations, including Graph Cellular Automata (GCA) and their integration with Graph Neural Networks (GNNs), which mark a paradigm shift from rule analysis to rule synthesis. By comparing these methods, we identify a significant gap between the representation of local rules and the analysis of global dynamics. This review concludes by proposing future research directions focused on developing hybrid graph models that can bridge this micro-macro divide, thereby providing more powerful tools for the study of complex systems.

### 1. Introduction

Cellular Automata (CA) are discrete dynamical systems in which space, time, and states are discrete.1 First conceptualized by Stanislaw Ulam and John von Neumann, they consist of a regular grid of cells, each with a finite state, that evolves in synchronous time steps according to a local rule dependent on the states of its neighbors.2 The simplest and most widely studied class is the Elementary Cellular Automaton (ECA), a one-dimensional CA with two possible states (0 or 1) where the update rule depends only on a cell and its immediate left and right neighbors.3 Despite their structural simplicity, ECAs can produce a remarkable spectrum of behaviors, from stable homogeneity to chaos and complex, self-organizing structures capable of universal computation.1

The ability to analyze, classify, and harness this complexity is contingent on the methods used to represent the underlying rules and their dynamics. Graph theory provides a powerful and intuitive framework for this purpose, offering a visual and structural language to describe the relationships between states and the transitions between them.5 However, a fundamental tension exists in this representation: should the graph depict the macroscopic evolution of the entire system's configuration space, or should it encode the microscopic logic of the local rule itself?

This review systematically explores this dichotomy by examining the predominant graph-based representations in ECA literature. We will first establish a foundational taxonomy of the 256 ECA rules, based on both their geometric symmetries and their emergent dynamical behaviors, which provides the necessary context for what these graphs aim to represent. We then delve into the two classical approaches: State Transition Graphs (STGs), which map the global dynamics, and De Bruijn Graphs (DBGs), which encode the local rules. For each, we analyze its construction, utility, and critical deficiencies. Subsequently, we discuss modern generalizations that extend the CA paradigm to arbitrary networks, namely Graph Cellular Automata (GCA) and the machine learning-driven Graph Neural Cellular Automata (GNCA). Finally, we present a comparative analysis of these methods and propose future research directions aimed at creating integrated representations that bridge the gap between local rule structure and global emergent complexity.

### 2. A Foundational Taxonomy of Elementary Cellular Automata Rules

Before analyzing how graphs represent ECA rules, it is essential to understand how the rules themselves are defined and classified. This taxonomy provides the framework for evaluating the expressive power and utility of different graph models.

#### 2.1 The 256 Rules and Wolfram Coding

An ECA consists of a 1D array of cells, each in state 0 or 1. A cell's next state is determined by its current state and the states of its two immediate neighbors.3 This 3-cell neighborhood has $2^3 = 8$ possible configurations (from 111 to 000). An ECA rule must specify a binary output for each of these 8 inputs, leading to a total of $2^8 = 256$ possible rules.3 Stephen Wolfram proposed a standard convention where the 8 outputs, ordered by the descending binary value of the input neighborhood, form an 8-bit binary number. This number's decimal equivalent, from 0 to 255, serves as the rule's unique identifier, known as its Wolfram Code.4 For example, the binary string 00011110 corresponds to the well-known chaotic Rule 30.4

#### 2.2 Classification by Geometric Symmetries

While there are 256 distinct rules, many are dynamically equivalent under simple geometric transformations.3 The two primary transformations are:

* **Reflection (Mirror Symmetry):** The rule is transformed by flipping the input neighborhoods left-to-right. A rule that is identical to its reflection is termed "amphichiral".3
* **Complementation (Color Inversion):** The rule is transformed by swapping all 0s and 1s in both the input neighborhoods and their outputs.3

Applying these symmetries partitions the 256 rules into equivalence classes. All rules within a class exhibit structurally identical spatio-temporal patterns, merely reflected or inverted. This process reduces the entire rule space to just **88 non-equivalent classes**, providing a minimal set for comprehensive analysis.3

#### 2.3 Classification by Dynamical Behavior

A more profound classification, proposed by Wolfram, is based on the qualitative nature of the behavior that emerges from a random initial state.1 This scheme categorizes rules into four classes:

* **Class 1 (Uniform):** Evolves rapidly to a spatially uniform, stable state (e.g., all 0s or all 1s). Examples include Rule 0 and Rule 32.3
* **Class 2 (Periodic):** Evolves to simple, stable, or oscillating periodic structures. Examples include Rule 4 and Rule 108.3
* **Class 3 (Chaotic):** Exhibits pseudo-random, aperiodic behavior where local changes propagate indefinitely. Examples include Rule 30 and Rule 90.3
* **Class 4 (Complex):** Produces complex localized structures (e.g., "gliders") that move and interact in non-trivial ways. These rules are considered to exist at the "edge of chaos" and are often capable of universal computation. Rule 54 and Rule 110 are the canonical examples.3

While this classification is qualitative, more rigorous quantitative methods based on properties like **limit cycle length** 15 and **algebraic semigroup theory** 16 have been developed, further refining our understanding of the complexity spectrum within ECAs.

### 3. State Transition Graphs: A Global View of System Dynamics

The most direct way to represent the complete behavior of a finite dynamical system is to map its entire phase space. The State Transition Graph (STG) achieves this by treating each possible global state of the system as a node.17

#### 3.1 Construction and Principles

For an ECA on a finite lattice of $N$ cells with periodic boundary conditions, there are $2^N$ possible global configurations. In an STG, each of these $2^N$ configurations is represented by a unique node.17 A directed edge is drawn from a node *A* to a node *B* if the global configuration *A* evolves into configuration *B* in a single time step under the ECA rule.17 Because the evolution of an ECA is deterministic, every configuration has exactly one successor. Consequently, every node in the STG has an out-degree of precisely one.17

#### 3.2 Topological Analysis of Dynamics

The topology of the STG is a direct visualization of the system's global dynamics, revealing its long-term behavior without the need for simulation.20

* **Attractors and Cycles:** Since the state space is finite, any evolutionary trajectory must eventually repeat, entering a periodic orbit. These orbits appear in the STG as directed **cycles**. The cycles are the system's **attractors**; once a state enters a cycle, it can never leave.17
* **Basins of Attraction and Transients:** The set of all states that eventually evolve into a particular attractor forms its **basin of attraction**. In the STG, this corresponds to a set of directed trees whose roots lie on the nodes of a cycle. The paths from the leaves of these trees to the cycle represent the **transient** phase of the dynamics before settling into a periodic behavior.17
* **"Garden of Eden" States:** ECA evolution is generally irreversible, meaning a configuration can have multiple predecessors or none at all. Nodes with an in-degree of zero represent **"Garden of Eden"** configurations—states that can only exist as initial conditions and can never be reached through evolution.17

#### 3.3 Deficiencies and Limitations

The primary and fatal flaw of the STG is the **state-space explosion**. The number of nodes, $2^N$, grows exponentially with the system size $N$. This makes the construction, storage, and analysis of the full STG computationally intractable for even modestly sized systems (e.g., $N > 20$).20 This severe scalability issue renders the STG a powerful theoretical concept but an impractical tool for most applications.

#### 3.4 Directions for Improvement

Directly improving the scalability of the STG is impossible due to its definition. However, research has shifted towards analyzing its properties without explicit construction. One powerful technique is the study of the graph's **automorphisms** (symmetries). By analyzing the symmetries of the global evolution function, researchers can classify ECAs based on how the number of automorphisms scales with system size, successfully identifying linear and chaotic rules through their distinct symmetry profiles.12 This approach abstracts key dynamical properties away from the unwieldy graph structure itself.

### 4. De Bruijn Graphs: Encoding the Local Rule Structure

To overcome the exponential complexity of STGs, an alternative representation was adopted from formal language theory: the De Bruijn Graph (DBG). Instead of mapping the global state space, the DBG provides a compact, fixed-size representation of the local rule itself.21

#### 4.1 Construction and Edge-Centric Encoding

For an ECA with neighborhood radius $r=1$, the DBG is constructed as follows 5:

* **Nodes:** The nodes represent all possible overlapping partial neighborhoods of length $2r=2$. For a binary alphabet, these are the four nodes: 00, 01, 10, and 11.
* **Edges:** A directed edge exists from node uv to vw if the last symbol of the source node matches the first symbol of the target node. Each edge thus uniquely represents a complete 3-cell neighborhood uvw. For example, an edge from 10 to 01 represents the neighborhood 101.
* **Edge Labels:** The core innovation lies in the use of edge labels to store data. The output of the local rule for the neighborhood represented by an edge is assigned as that edge's **label**.5 For Rule 110, where $f(101) \to 1$, the edge from 10 to 01 would be labeled 1.

In this model, the edge becomes the fundamental unit of computation, encoding a single application of the local rule. A path of length $L$ through the graph represents a configuration of length $L+2$, and the sequence of edge labels along that path gives the corresponding evolved configuration for the central $L$ cells.21

#### 4.2 Applications in Algorithmic Analysis

The DBG's fixed and small size makes it an ideal data structure for algorithmically determining a rule's intrinsic properties without simulation.22 Properties like **surjectivity** (whether Garden of Eden states exist) and **reversibility** (whether every configuration has a unique predecessor) can be decided in polynomial time by analyzing paths and reachability on the DBG and its derivatives.25

#### 4.3 Deficiencies and Limitations

The DBG's main deficiency is the inverse of the STG's: it completely **discards information about global dynamics**. The DBG for a chaotic rule like Rule 30 is structurally identical to the DBG for a simple periodic rule like Rule 4; their differences are confined to the edge labels.5 The graph's topology alone reveals nothing about the attractors, transients, or complex patterns that emerge over time. It is a tool for analyzing the *rule*, not the *dynamics*.

#### 4.4 Directions for Improvement

The path for improvement involves building more sophisticated graph structures upon the DBG foundation. To solve complex problems like reversibility, researchers have developed extensions such as the **Pair Graph** (whose nodes are pairs of DBG nodes) and the **Subset Graph** (whose nodes are sets of DBG nodes).5 These extensions transform properties like non-reversibility into standard graph-theoretic problems like reachability. A promising future direction is to create **annotated DBGs**, where nodes or edges are decorated with statistical metrics (e.g., transition probabilities, local entropy) derived from large-scale simulations. This could re-introduce a measure of dynamical information into the compact rule representation, partially bridging the gap with STGs.

### 5. Modern Generalizations: Cellular Automata on Arbitrary Graphs

Classical CA are defined on regular, grid-like structures. However, many real-world systems, from social networks to biological pathways, are better described by irregular graphs. This has motivated the generalization of the CA model to arbitrary network topologies.

#### 5.1 Graph Cellular Automata (GCA)

In a Graph Cellular Automaton (GCA), the cells are the vertices of an arbitrary graph, and a cell's neighborhood is defined by its adjacent vertices.26 This generalization allows the modeling of local dynamics on complex networks, enabling the study of how network topology (e.g., degree distribution, modularity) influences functional outcomes like information propagation or pattern formation.28 A key challenge is defining the local rule, as neighborhoods are no longer ordered and vary in size. Common solutions include **totalistic rules**, which depend only on the sum of neighbor states 28, or **anisotropic rules** that leverage edge attributes.26

#### 5.2 The Rise of Graph Neural Cellular Automata (GNCA)

The emergence of Graph Neural Networks (GNNs) has provided a powerful new framework for studying GCA.26 The message-passing mechanism in GNNs, where a node updates its state by aggregating information from its neighbors, is a direct analogue of a CA's local update rule.26 This allows a GCA's transition function to be parameterized by a GNN.

#### 5.3 A Paradigm Shift from Analysis to Synthesis

This GNN-based approach, termed Graph Neural Cellular Automata (GNCA), represents a fundamental paradigm shift. Instead of being given a fixed rule to *analyze*, a GNCA can be given a target global behavior and trained to *learn* or *synthesize* the local rule that produces it.26 For example, a GNCA can learn rules for regeneration, pattern formation, or simulating physical processes on a graph. This moves the field from studying a fixed set of 256 rules to designing bespoke rules for specific tasks and topologies.

#### 5.4 Deficiencies and Future Directions

The primary drawback of the GCA/GNCA framework is the loss of the analytical simplicity and tractability found in 1D ECAs. The behavior is highly dependent on both the learned rule and the underlying graph topology, making formal analysis difficult. The direction for improvement lies in designing more principled GNN architectures. For instance, using **E(n)-equivariant GNNs** can enforce physical symmetries like rotation and translation, leading to more robust and generalizable learned rules that are inherently isotropic.29

### 6. Comparative Analysis and Future Research Directions

The three major graph representation paradigms—STG, DBG, and GCA/GNCA—offer distinct perspectives on cellular automata, each with a unique trade-off between descriptive power and computational feasibility.

#### 6.1 A Synthesis of Methods

The following table summarizes the core differences between these approaches:

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **State Transition Graph (STG)** | **De Bruijn Graph (DBG)** | **Graph Neural CA (GNCA)** |
| **Representation Unit** | Global System Configuration | Local Rule Neighborhood | Node State on an Arbitrary Graph |
| **Primary Use Case** | Theoretical analysis of global dynamics | Algorithmic analysis of rule properties | Learning/synthesizing rules for a target behavior |
| **Computational Scale** | Exponential ($O(2^N)$) | Constant (for ECA) | Polynomial in graph size ($ |
| **Meaning of an Edge** | A single global time step ($C\_t \to C\_{t+1}$) | A single local computation ($f(l,c,r) \to s'$) | An interaction channel in the underlying network |
| **Key Strength** | Complete description of phase space | Compact, fixed-size, algorithmically efficient | Generalizable to any topology; enables rule design |
| **Key Weakness** | Computationally intractable | Devoid of global dynamic information | Lacks analytical tractability; a "black box" |

#### 6.2 Bridging the Micro-Macro Gap

The central challenge highlighted by this review is the profound disconnect between the microscopic representation of local rules (DBG) and the macroscopic representation of emergent global dynamics (STG). An ideal representation would be both computationally tractable and dynamically informative, but current methods force a choice between one or the other. Closing this "micro-macro" gap remains the foremost open problem in the field.

#### 6.3 Recommendations for Future Research

Based on this analysis, we propose three promising avenues for future investigation:

1. **Develop Hybrid or "Meso-Scale" Graph Models:** A significant innovation would be the creation of a graph representation that operates at an intermediate scale. In such a model, nodes would not represent individual neighborhoods (too small) or entire configurations (too large), but rather recurring patterns, motifs, or other meaningful sub-structures. Edges would then represent the probabilistic or deterministic rules governing the transitions between these meso-scale patterns, offering a compressed yet dynamically relevant view of the system.
2. **Create Dynamically Annotated De Bruijn Graphs:** The DBG provides a robust and efficient foundation. Its primary limitation—the lack of dynamic information—could be mitigated by augmenting it. One could annotate the nodes or edges of a DBG with metrics derived from extensive simulations, such as traversal frequency, contribution to global entropy, or correlation with the emergence of complex structures. This would embed empirical knowledge of the dynamics directly onto the static graph of the rule.
3. **Apply GNNs for ECA Classification:** While GNCAs are typically used for synthesis on arbitrary graphs, the GNN framework could be repurposed as a powerful analytical tool for classical ECAs. A GNN could be trained on a dataset of DBGs labeled with their corresponding Wolfram class or other dynamic properties (e.g., limit cycle length). The goal would be to learn a function that can predict the emergent global behavior of a rule simply by inspecting the structure and labeling of its DBG, effectively learning the mapping from local rule to global complexity.

### 7. Conclusion

The study of Elementary Cellular Automata through the lens of graph theory reveals a rich and evolving landscape of representative techniques. The journey has taken us from the theoretically elegant but practically infeasible State Transition Graphs, which capture every nuance of global dynamics, to the computationally efficient and algorithmically powerful De Bruijn Graphs, which perfectly encode local rules at the cost of dynamic insight. The recent advent of Graph Neural Cellular Automata has opened a new frontier, shifting the focus from analyzing a fixed set of rules to synthesizing novel ones for specific tasks on arbitrary networks.

Despite these advances, the fundamental challenge of creating a single, tractable representation that unifies local rules with their global emergent consequences persists. The most fertile ground for future research lies in bridging this micro-macro divide. The development of hybrid, meso-scale, or dynamically-aware graph models promises not only a deeper understanding of cellular automata but also more potent tools for analyzing and designing complex systems across all scientific domains.