Cellular Automata

Models for a discrete world

Contents

Articles

References

Article Licenses

Cellular automaton

Gosper's Glider Gun creating "gliders" in the cellular automaton Conway's Game of Life[1]

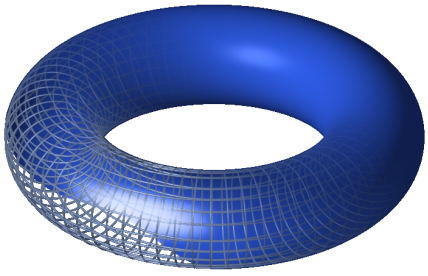
A **cellular automaton** (pl. **cellular automata**, abbrev. **CA**) is a discrete model studied in computability theory, mathematics, physics, complexity science, theoretical biology and microstructure modeling. It consists of a regular grid of *cells*, each in one of a finite number of *states*, such as "On" and "Off" (in contrast to a coupled map lattice). The grid can be in any finite number of dimensions. For each cell, a set of cells called its *neighborhood* (usually including the cell itself) is defined relative to the specified cell. For example, the neighborhood of a cell might be defined as the set of cells a distance of 2 or less from the cell. An initial state (time t=0) is selected by assigning a state for each cell. A new *generation* is created (advancing t by 1), according to some fixed rule (generally, a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. For example, the rule might be that the cell is "On" in the next generation if exactly two of the cells in the neighborhood are "On" in the current generation, otherwise the cell is "Off" in the next generation. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously, though exceptions are known.

Cellular automata are also called "cellular spaces", "tessellation automata", "homogeneous structures", "cellular structures", "tessellation structures", and "iterative arrays".[2]

Overview

One way to simulate a two-dimensional cellular automaton is with an infinite sheet of graph paper along with a set of rules for the cells to follow. Each square is called a "cell" and each cell has two possible states, black and white. The "neighbors" of a cell are the 8 squares touching it. For such a cell and its neighbors, there are 512 (= 2^9) possible patterns. For each of the 512 possible patterns, the rule table would state whether the center cell will be black or white on the next time interval. Conway's Game of Life is a popular version of this model.

It is usually assumed that every cell in the universe starts in the same state, except for a finite number of cells in other states, often called a *configuration*. More generally, it is sometimes assumed that the universe starts out covered with a periodic pattern, and only a finite number of cells violate that pattern. The latter assumption is common in one-dimensional cellular automata.



A torus, a toroidal shape.

Cellular automata are often simulated on a finite grid rather than an infinite one. In two dimensions, the universe would be a rectangle instead of an infinite plane. The obvious problem with finite grids is how to handle the cells on the edges. How they are handled will affect the values of all the cells in the grid. One possible method is to allow the values in those cells to remain constant. Another method is to define neighbourhoods differently for these cells. One could say that they have fewer neighbours, but then one would also have to define new rules for the cells located on the edges. These cells are usually handled with a *toroidal* arrangement: when one goes off the top, one comes in at the corresponding position on the bottom, and when one goes off the left, one comes in on the right. (This essentially simulates an infinite periodic tiling, and in the field of

partial differential equations is sometimes referred to as *periodic* boundary conditions.) This can be visualized as taping the left and right edges of the rectangle to form a tube, then taping the top and bottom edges of the tube to form a torus (doughnut shape). Universes of other dimensions are handled similarly. This is done in order to solve boundary problems with neighborhoods, but another advantage of this system is that it is easily programmable using modular arithmetic functions. For example, in a 1-dimensional cellular automaton like the examples below, the neighborhood of a cell x_i^t—where t is the time step (vertical), and i is the index (horizontal) in one generation—is $\{x_{i-1}^{t-1}, x_i^{t-1}, x_{i+1}^{t-1}\}$. There will obviously be problems when a neighbourhood on a left border references its upper left cell, which is not in the cellular space, as part of its neighborhood.

History

Stanisław Ulam, while working at the Los Alamos National Laboratory in the 1940s, studied the growth of crystals, using a simple lattice network as his model. At the same time, John von Neumann, Ulam's colleague at Los Alamos, was working on the problem of self-replicating systems. Von Neumann's initial design was founded upon the notion of one robot building another robot. This design is known as the kinematic model.[3] [4] As he developed this design, von Neumann came to realize the great difficulty of building a self-replicating robot, and of the great cost in providing the robot with a "sea of parts" from which to build its replicant. Ulam suggested that von Neumann develop his design around a mathematical abstraction, such as the one Ulam used to study crystal growth. Thus was born the first system of cellular automata. Like Ulam's lattice network, von Neumann's cellular automata are two-dimensional, with his self-replicator implemented algorithmically. The result was a universal copier and constructor working within a CA with a small neighborhood (only those cells that touch are neighbors; for von Neumann's cellular automata, only orthogonal cells), and with 29 states per cell. Von Neumann gave an existence proof that a particular pattern would make endless copies of itself within the given cellular universe. This design is known as the tessellation model, and is called a von Neumann universal constructor.

Also in the 1940s, Norbert Wiener and Arturo Rosenblueth developed a cellular automaton model of excitable media.[5] Their specific motivation was the mathematical description of impulse conduction in cardiac systems. Their original work continues to be cited in modern research publications on cardiac arrhythmia and excitable systems.[6]

In the 1960s, cellular automata were studied as a particular type of dynamical system and the connection with the mathematical field of symbolic dynamics was established for the first time. In 1969, Gustav A. Hedlund compiled many results following this point of view[7] in what is still considered as a seminal paper for the mathematical study of cellular automata. The most fundamental result is the characterization in the Curtis–Hedlund–Lyndon theorem of the set of global rules of cellular automata as the set of continuous endomorphisms of shift spaces.

In the 1970s a two-state, two-dimensional cellular automaton named Game of Life became very widely known, particularly among the early computing community. Invented by John Conway and popularized by Martin Gardner in a *Scientific American* article,[8] its rules are as follows: If a cell has 2 black neighbours, it stays the same. If it has 3 black neighbours, it becomes black. In all other situations it becomes white. Despite its simplicity, the system achieves an impressive diversity of behavior, fluctuating between apparent randomness and order. One of the most apparent features of the Game of Life is the frequent occurrence of *gliders*, arrangements of cells that essentially move themselves across the grid. It is possible to arrange the automaton so that the gliders interact to perform computations, and after much effort it has been shown that the Game of Life can emulate a universal Turing machine.[9] Possibly because it was viewed as a largely recreational topic, little follow-up work was done outside of investigating the particularities of the Game of Life and a few related rules.

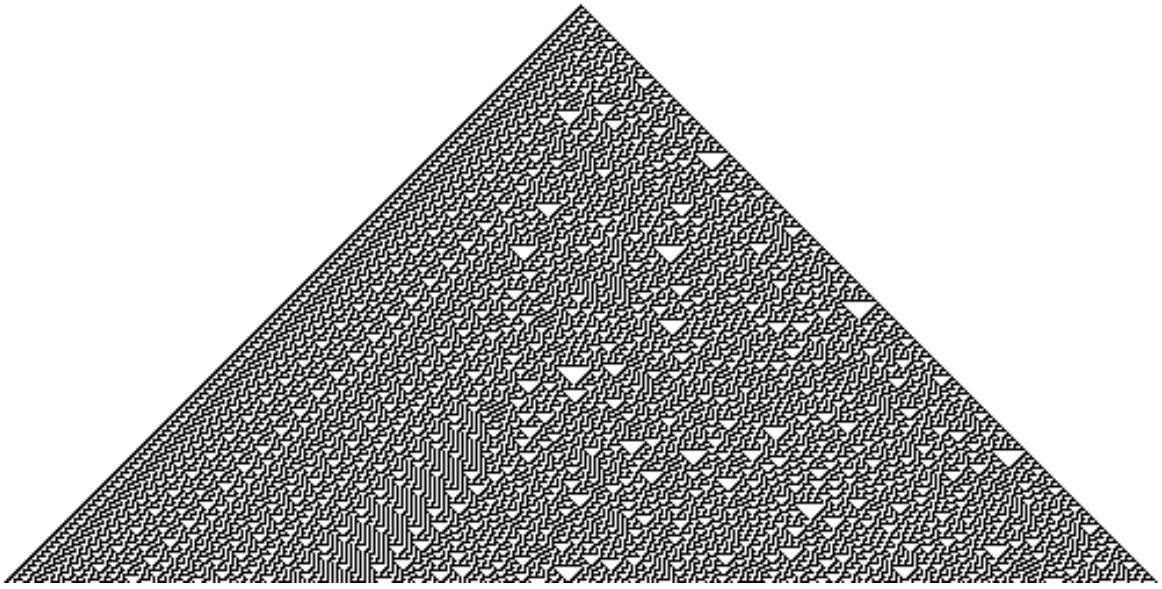
In 1969, however, German computer pioneer Konrad Zuse published his book *Calculating Space*, proposing that the physical laws of the universe are discrete by nature, and that the entire universe is the output of a deterministic computation on a giant cellular automaton. This was the first book on what today is called digital physics.

In 1983 Stephen Wolfram published the first of a series of papers systematically investigating a very basic but essentially unknown class of cellular automata, which he terms *elementary cellular automata* (see below). The unexpected complexity of the behavior of these simple rules led Wolfram to suspect that complexity in nature may be due to similar mechanisms. Additionally, during this period Wolfram formulated the concepts of intrinsic randomness and computational irreducibility, and suggested that rule 110 may be universal—a fact proved later by Wolfram's research assistant Matthew Cook in the 1990s.

In 2002 Wolfram published a 1280-page text *A New Kind of Science*, which extensively argues that the discoveries about cellular automata are not isolated facts but are robust and have significance for all disciplines of science. Despite much confusion in the press and academia, the book did not argue for a fundamental theory of physics based on cellular automata, and although it did describe a few specific physical models based on cellular automata, it also provided models based on qualitatively different abstract systems.

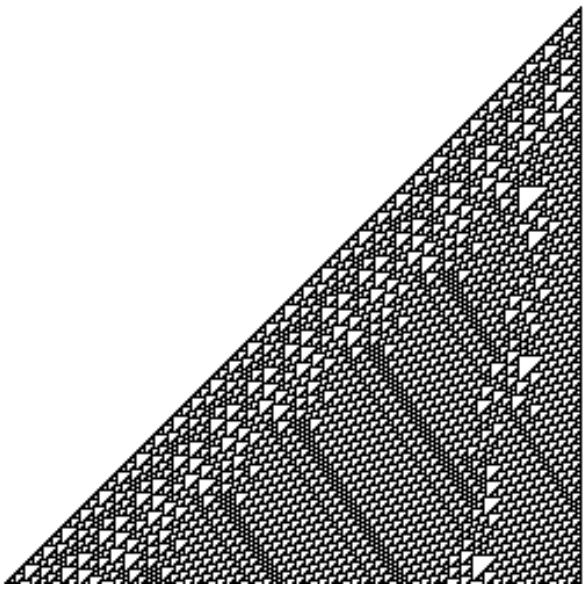
Elementary cellular automata

The simplest nontrivial CA would be one-dimensional, with two possible states per cell, and a cell's neighbors defined to be the adjacent cells on either side of it. A cell and its two neighbors form a neighborhood of 3 cells, so there are 2^3=8 possible patterns for a neighborhood. A rule consists of deciding, for each pattern, whether the cell will be a 1 or a 0 in the next generation. There are then 2^8=256 possible rules. These 256 CAs are generally referred to by their Wolfram code, a standard naming convention invented by Stephen Wolfram which gives each rule a number from 0 to 255. A number of papers have analyzed and compared these 256 CAs. The rule 30 and rule 110 CAs are particularly interesting. The images below show the history of each when the starting configuration consists of a 1 (at the top of each image) surrounded by 0's. Each row of pixels represents a generation in the history of the automation, with *t*=0 being the top row. Each pixel is colored white for 0 and black for 1.



Rule 30 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0



Rule 110 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Rule 30 exhibits *class 3* behavior, meaning even simple input patterns such as that shown lead to chaotic, seemingly random histories.

Rule 110, like the Game of Life, exhibits what Wolfram calls *class 4* behavior, which is neither completely random nor completely repetitive. Localized structures appear and interact in various complicated-looking ways. In the course of the development of *A New Kind of Science*, as a research assistant to Stephen Wolfram in 1994, Matthew Cook proved that some of these structures were rich enough to support universality. This result is interesting because rule 110 is an extremely simple one-dimensional system, and one which is difficult to engineer to perform specific behavior. This result therefore provides significant support for Wolfram's view that class 4 systems are inherently likely to be universal. Cook presented his proof at a Santa Fe Institute conference on Cellular Automata in 1998, but Wolfram blocked the proof from being included in the conference proceedings, as Wolfram did not want the proof to be announced before the publication of *A New Kind of Science*. In 2004, Cook's proof was finally published in Wolfram's journal Complex Systems [10] (Vol. 15, No. 1), over ten years after Cook came up with it. Rule 110 has been the basis over which some of the smallest universal Turing machines have been built, inspired on the breakthrough concepts that the development of the proof of rule 110 universality produced.

Reversible

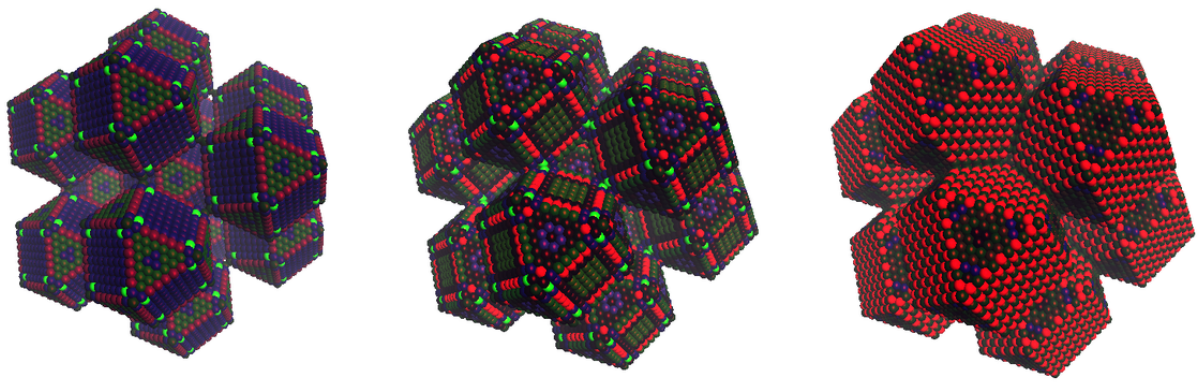
A cellular automaton is said to be *reversible* if for every current configuration of the cellular automaton there is exactly one past configuration (preimage). If one thinks of a cellular automaton as a function mapping configurations to configurations, reversibility implies that this function is bijective. If a cellular automaton is reversible, its time-reversed behavior can also be described as a cellular automaton; this fact is a consequence of the Curtis–Hedlund–Lyndon theorem, a topological characterization of cellular automata.[11] [12] For cellular automata in which not every configuration has a preimage, the configurations without preimages are called *Garden of Eden patterns*.

For one dimensional cellular automata there are known algorithms for deciding whether a rule is reversible or irreversible.[13] [14] However, for cellular automata of two or more dimensions reversibility is undecidable; that is, there is no algorithm that takes as input an automaton rule and is guaranteed to determine correctly whether the automaton is reversible. The proof by Jarkko Kari is related to the tiling problem by Wang tiles.[15]

Reversible CA are often used to simulate such physical phenomena as gas and fluid dynamics, since they obey the laws of thermodynamics. Such CA have rules specially constructed to be reversible. Such systems have been studied by Tommaso Toffoli, Norman Margolus and others. Several techniques can be used to explicitly construct reversible CA with known inverses. Two common ones are the second order cellular automaton and the block cellular automaton, both of which involve modifying the definition of a CA in some way. Although such automata do not strictly satisfy the definition given above, it can be shown that they can be emulated by conventional CAs with sufficiently large neighborhoods and numbers of states, and can therefore be considered a subset of conventional CA. Conversely, it has been shown that every reversible cellular automaton can be emulated by a block cellular automaton.[16]

Totalistic

A special class of CAs are *totalistic* CAs. The state of each cell in a totalistic CA is represented by a number (usually an integer value drawn from a finite set), and the value of a cell at time t depends only on the *sum* of the values of the cells in its neighborhood (possibly including the cell itself) at time $t-1$.[17] [18] If the state of the cell at time t does depend on its own state at time $t-1$ then the CA is properly called *outer totalistic*.[18] Conway's Game of Life is an example of an outer totalistic CA with cell values 0 and 1; outer totalistic cellular automata with the same Moore neighborhood structure as Life are sometimes called life-like cellular automata.[19]



3D totalistic cellular automata

Classification

Stephen Wolfram, in *A New Kind of Science* and in several papers dating from the mid-1980s, defined four classes into which cellular automata and several other simple computational models can be divided depending on their behavior. While earlier studies in cellular automata tended to try to identify type of patterns for specific rules, Wolfram's classification was the first attempt to classify the rules themselves. In order of complexity the classes are:

- Class 1: Nearly all initial patterns evolve quickly into a stable, homogeneous state. Any randomness in the initial pattern disappears.
- Class 2: Nearly all initial patterns evolve quickly into stable or oscillating structures. Some of the randomness in the initial pattern may filter out, but some remains. Local changes to the initial pattern tend to remain local.
- Class 3: Nearly all initial patterns evolve in a pseudo-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise. Local changes to the initial pattern tend to spread indefinitely.
- Class 4: Nearly all initial patterns evolve into structures that interact in complex and interesting ways. Class 2 type stable or oscillating structures may be the eventual outcome, but the number of steps required to reach this state may be very large, even when the initial pattern is relatively simple. Local changes to the initial pattern may

spread indefinitely. Wolfram has conjectured that many, if not all class 4 cellular automata are capable of universal computation. This has been proven for Rule 110 and Conway's game of Life.

These definitions are qualitative in nature and there is some room for interpretation. According to Wolfram, "...with almost any general classification scheme there are inevitably cases which get assigned to one class by one definition and another class by another definition. And so it is with cellular automata: there are occasionally rules...that show some features of one class and some of another."[20] Wolfram's classification has been empirically matched to a clustering of the compressed lengths of the outputs of cellular automata.[21]

There have been several attempts to classify CA in formally rigorous classes, inspired by the Wolfram's classification. For instance, Culik and Yu proposed three well-defined classes (and a fourth one for the automata not matching any of these), which are sometimes called Culik-Yu classes; membership in these proved to be undecidable.[22] [23] [24]

Evolving cellular automata using genetic algorithms

Recently there has been a keen interest in building decentralized systems, be they sensor networks or more sophisticated micro level structures designed at the network level and aimed at decentralized information processing. The idea of emergent computation came from the need of using distributed systems to do information processing at the global level.[25] The area is still in its infancy, but some people have started taking the idea seriously. Melanie Mitchell who is Professor of Computer Science at Portland State University and also Santa Fe Institute External Professor[26] has been working on the idea of using self-evolving cellular arrays to study emergent computation and distributed information processing.[25] Mitchell and colleagues are using evolutionary computation to program cellular arrays.[27] Computation in decentralized systems is very different from classical systems, where the information is processed at some central location depending on the system's state. In decentralized system, the information processing occurs in the form of global and local pattern dynamics.

The inspiration for this approach comes from complex natural systems like insect colonies, nervous system and economic systems.[27] The focus of the research is to understand how computation occurs in an evolving decentralized system. In order to model some of the features of these systems and study how they give rise to emergent computation, Mitchell and collaborators at the SFI have applied Genetic Algorithms to evolve patterns in cellular automata. They have been able to show that the GA discovered rules that gave rise to sophisticated emergent computational strategies.[28] Mitchell's group used a single dimensional binary array where each cell has six neighbors. The array can be thought of as a circle where the first and last cells are neighbors. The evolution of the array was tracked through the number of ones and zeros after each iteration. The results were plotted to show clearly how the network evolved and what sort of emergent computation was visible.

The results produced by Mitchell's group are interesting, in that a very simple array of cellular automata produced results showing coordination over global scale, fitting the idea of emergent computation. Future work in the area may include more sophisticated models using cellular automata of higher dimensions, which can be used to model complex natural systems.

Cryptography use

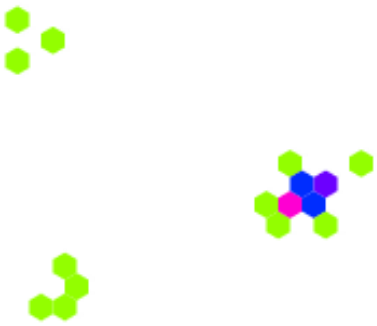
Rule 30 was originally suggested as a possible Block cipher for use in cryptography (See CA-1.1).

Cellular automata have been proposed for public key cryptography. The one way function is the evolution of a finite CA whose inverse is believed to be hard to find. Given the rule, anyone can easily calculate future states, but it appears to be very difficult to calculate previous states. However, the designer of the rule can create it in such a way as to be able to easily invert it. Therefore, it is apparently a trapdoor function, and can be used as a public-key cryptosystem. The security of such systems is not currently known.

Related automata

There are many possible generalizations of the CA concept.

One way is by using something other than a rectangular (cubic, *etc.*) grid. For example, if a plane is tiled with regular hexagons, those hexagons could be used as cells. In many cases the resulting cellular automata are equivalent to those with rectangular grids with specially designed neighborhoods and rules.



A cellular automaton based on hexagonal cells instead of squares (rule 34/2).

Also, rules can be probabilistic rather than deterministic. A probabilistic rule gives, for each pattern at time *t*, the probabilities that the central cell will transition to each possible state at time *t*+1. Sometimes a simpler rule is used; for example: "The rule is the Game of Life, but on each time step there is a 0.001% probability that each cell will transition to the opposite color."

The neighborhood or rules could change over time or space. For example, initially the new state of a cell could be determined by the horizontally adjacent cells, but for the next generation the vertical cells would be used.

The grid can be finite, so that patterns can "fall off" the edge of the universe.

In CA, the new state of a cell is not affected by the new state of other cells. This could be changed so that, for instance, a 2 by 2 block of cells can be determined by itself and the cells adjacent to itself.

There are *continuous automata*. These are like totalistic CA, but instead of the rule and states being discrete (*e.g.* a table, using states {0,1,2}), continuous functions are used, and the states become continuous (usually values in [0,1]). The state of a location is a finite number of real numbers. Certain CA can yield diffusion in liquid patterns in this way.

Continuous spatial automata have a continuum of locations. The state of a location is a finite number of real numbers. Time is also continuous, and the state evolves according to differential equations. One important example is reaction-diffusion textures, differential equations proposed by Alan Turing to explain how chemical reactions could create the stripes on zebras and spots on leopards.[29] When these are approximated by CA, such CAs often yield similar patterns. MacLennan [30] considers continuous spatial automata as a model of computation.

There are known examples of continuous spatial automata which exhibit propagating phenomena analogous to gliders in the Game of Life.[31]

Biology

Conus textile exhibits a cellular automaton pattern on its shell

Some biological processes occur—or can be simulated—by cellular automata.

Patterns of some seashells, like the ones in *Conus* and *Cymbiola* genus, are generated by natural CA. The pigment cells reside in a narrow band along the shell's lip. Each cell secretes pigments according to the activating and inhibiting activity of its neighbour pigment cells, obeying a natural version of a mathematical rule. The cell band leaves the colored pattern on the shell as it grows slowly. For example, the widespread species *Conus textile* bears a pattern resembling Wolfram's rule 30 CA.

Plants regulate their intake and loss of gases via a CA mechanism. Each stoma on the leaf acts as a cell.[32]

Moving wave patterns on the skin of cephalopods can be simulated with a two-state, two-dimensional cellular automata, each state corresponding to either an expanded or retracted chromatophore.[33]

Threshold automata have been invented to simulate neurons, and complex behaviors such as recognition and learning can be simulated.

Fibroblasts bear similarities to cellular automata, as each fibroblast only interacts with its neighbors.[34]

Chemical types

The Belousov–Zhabotinsky reaction is a spatio-temporal chemical oscillator which can be simulated by means of a cellular automaton. In the 1950s A. M. Zhabotinsky (extending the work of B. P. Belousov) discovered that when a thin, homogenous layer of a mixture of malonic acid, acidified bromate, and a ceric salt were mixed together and left undisturbed, fascinating geometric patterns such as concentric circles and spirals propagate across the medium. In the "Computer Recreations" section of the August 1988 issue of *Scientific American*,[35] A. K. Dewdney discussed a cellular automaton[36] which was developed by Martin Gerhardt and Heike Schuster of the University of Bielefeld (West Germany). This automaton produces wave patterns resembling those in the Belousov-Zhabotinsky reaction.

Computer processors

CA processors are physical (not computer simulated) implementations of CA concepts, which can process information computationally. Processing elements are arranged in a regular grid of identical cells. The grid is usually a square tiling, or tessellation, of two or three dimensions; other tilings are possible, but not yet used. Cell states are determined only by interactions with adjacent neighbor cells. No means exists to communicate directly with cells farther away.

One such CA processor array configuration is the systolic array.

Cell interaction can be via electric charge, magnetism, vibration (phonons at quantum scales), or any other physically useful means. This can be done in several ways so no wires are needed between any elements.

This is very unlike processors used in most computers today, von Neumann designs, which are divided into sections with elements that can communicate with distant elements over wires.

Error correction coding

CA have been applied to design error correction codes in the paper "Design of CAECC – Cellular Automata Based Error Correcting Code", by D. Roy Chowdhury, S. Basu, I. Sen Gupta, P. Pal Chaudhuri. The paper defines a new scheme of building SEC-DED codes using CA, and also reports a fast hardware decoder for the code.

CA as models of the fundamental physical reality

As Andrew Ilachinski points out in his *Cellular Automata*, many scholars have raised the question of whether the universe is a cellular automaton.[37] Ilachinsky argues that the importance of this question may be better appreciated with a simple observation, which can be stated as follows. Consider the evolution of rule 110: if it were some kind of "alien physics", what would be a reasonable description of the observed patterns?[38] If you didn't know how the images were generated, you might end up conjecturing about the movement of some particle-like objects (indeed, physicist James Crutchfield made a rigorous mathematical theory out of this idea proving the statistical emergence of "particles" from CA).[39] Then, as the argument goes, one might wonder if *our* world, which is currently well described by physics with particle-like objects, could be a CA at its most fundamental level.

While a complete theory along this line is still to be developed, entertaining and developing this hypothesis led scholars to interesting speculation and fruitful intuitions on how can we make sense of our world within a discrete

framework. Marvin Minsky, the AI pioneer, investigated how to understand particle interaction with a four-dimensional CA lattice;[40] Konrad Zuse—the inventor of the first working computer, the Z3—developed an irregularly organized lattice to address the question of the information content of particles.[41] More recently, Edward Fredkin exposed what he terms the "finite nature hypothesis", i.e., the idea that "ultimately every quantity of physics, including space and time, will turn out to be discrete and finite."[42] Fredkin and Stephen Wolfram are strong proponents of a CA-based physics.

In recent years, other suggestions along these lines have emerged from literature in non-standard computation. Stephen Wolfram's *A New Kind of Science* considers CA to be the key to understanding a variety of subjects, physics included. The *Mathematics Of the Models of Reference*—created by iLabs[43] founder Gabriele Rossi and developed with Francesco Berto and Jacopo Tagliabue—features an original 2D/3D universe based on a new "rhombic dodecahedron-based" lattice and a unique rule. This model satisfies universality (it is equivalent to a Turing Machine) and perfect reversibility (a *desideratum* if one wants to conserve various quantities easily and never lose information), and it comes embedded in a first-order theory, allowing computable, qualitative statements on the universe evolution.[44]

In popular culture

- One-dimensional cellular automata were mentioned in the Season 2 episode of NUMB3RS "Better or Worse".[45]
- The Hacker Emblem, a symbol for hacker culture proposed by Eric S. Raymond, depicts a glider from Conway's Game of Life.[46]
- The Autoverse, an artificial life simulator in the novel *Permutation City*, is a cellular automaton.[47]
- Cellular automata are discussed in the novel *Bloom*.[48]
- Cellular automata are central to Robert J. Sawyer's trilogy *WWW* in an attempt to explain how Webmind spontaneously attained consciousness.[49]

Reference notes

[1] Daniel Dennett (1995), *Darwin's Dangerous Idea*, Penguin Books, London, ISBN 978-0-140-16734-4, ISBN 0-140-16734-X

[2] Wolfram, Stephen (1983), "Statistical mechanics of cellular automata" (http://www.stephenwolfram.com/publications/articles/ca/83-statistical/), *Reviews of Modern Physics* **55** (3): 601–644, Bibcode 1983RvMP...55..601W, doi:10.1103/RevModPhys.55.601,

[3] John von Neumann, "The general and logical theory of automata," in L.A. Jeffress, ed., Cerebral Mechanisms in Behavior – The Hixon Symposium, John Wiley & Sons, New York, 1951, pp. 1-31.

[4] John G. Kemeny, "Man viewed as a machine," Sci. Amer. 192(April 1955):58-67; Sci. Amer. 192(June 1955):6 (errata).

[5] Wiener, N.; Rosenblueth, A. (1946). "The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle". *Arch. Inst. Cardiol. México* **16**: 205.

[6] Davidenko, J. M.; Pertsov, A. V.; Salomonsz, R.; Baxter, W.; Jalife, J. (1992). "Stationary and drifting spiral waves of excitation in isolated cardiac muscle". *Nature* **355** (6358): 349–351. Bibcode 1992Natur.355..349D. doi:10.1038/355349a0. PMID 1731248.

[7] Hedlund, G. A. (1969). "Endomorphisms and automorphisms of the shift dynamical system" (http://www.springerlink.com/content/k629151862130377/). *Math. Systems Theory* **3** (4): 320–3751. doi:10.1007/BF01691062. .

[8] Gardner, M. (1970). "MATHEMATICAL GAMES The fantastic combinations of John Conway's new solitaire game "life"" (http://www.ibiblio.org/lifepatterns/october1970.html). *Scientific American*: 120–123. .

[9] Paul Chapman. Life universal computer. http://www.igblan.free-online.co.uk/igblan/ca/November 2002

[10] http://www.complex-systems.com

[11] Richardson, D. (1972), "Tessellations with local transformations", *J. Computer System Sci.* **6** (5): 373–388, doi:10.1016/S0022-0000(72)80009-6.

[12] Margenstern, Maurice (2007), *Cellular Automata in Hyperbolic Spaces - Tome I, Volume 1* (http://books.google.com/books?id=wGjX1PpFqjAC&pg=PA134), Archives contemporaines, p. 134, ISBN 9782847030334, .

[13] Serafino Amoroso, Yale N. Patt, *Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures*. J. Comput. Syst. Sci. 6(5): 448-464 (1972)

[14] Sutner, Klaus (1991), "De Bruijn Graphs and Linear Cellular Automata" (http://www.complex-systems.com/pdf/05-1-3.pdf), *Complex Systems* **5**: 19–30, .

[15] Kari, Jarkko (1990), "Reversibility of 2D cellular automata is undecidable", *Physica D* **45**: 379–385, Bibcode 1990PhyD...45..379K, doi:10.1016/0167-2789(90)90195-U.

[16] Kari, Jarkko (1999), "On the circuit depth of structurally reversible cellular automata", *Fundamenta Informaticae* **38**: 93–107; Durand-Lose, Jérôme (2001), "Representing reversible cellular automata with reversible block cellular automata" (http://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/download/264/855), *Discrete Mathematics and Theoretical Computer Science* **AA**: 145–154, .

[17] Stephen Wolfram, *A New Kind of Science*, p. 60 (http://www.wolframscience.com/nksonline/page-0060-text).

[18] Ilachinski, Andrew (2001), *Cellular automata: a discrete universe* (http://books.google.com/books?id=3Hx2lx_pEF8C&pg=PA4), World Scientific, pp. 44–45, ISBN 9789812381835, .

[19] The phrase "life-like cellular automaton" dates back at least to Barral, Chaté & Manneville (1992), who used it in a broader sense to refer to outer totalistic automata, not necessarily of two dimensions. The more specific meaning given here was used e.g. in several chapters of Adamatzky (2010). See: Barral, Bernard; Chaté, Hugues; Manneville, Paul (1992), "Collective behaviors in a family of high-dimensional cellular automata", *Physics Letters A* **163** (4): 279–285, doi:10.1016/0375-9601(92)91013-H; Adamatzky, Andrew, ed. (2010), *Game of Life Cellular Automata*, Springer, ISBN 9781849962162.

[20] Stephen Wolfram, *A New Kind of Science* p231 ff.

[21] Hector Zenil, *Compression-based investigation of the dynamical properties of cellular automata and other systems* journal of Complex Systems 19:1, 2010

[22] G. Cattaneo, E. Formenti, L. Margara (1998). "Topological chaos and CA" (http://books.google.com/books?id=dGs87s5Pft0C&pg=PA239). In M. Delorme, J. Mazoyer. *Cellular automata: a parallel model*. Springer. p. 239. ISBN 9780792354932. .

[23] Burton H. Voorhees (1996). *Computational analysis of one-dimensional cellular automata* (http://books.google.com/books?id=WcZTQHPrG68C&pg=PA8). World Scientific. p. 8. ISBN 9789810222215. .

[24] Max Garzon (1995). *Models of massive parallelism: analysis of cellular automata and neural networks*. Springer. p. 149. ISBN 9783540561491.

[25] The Evolution of Emergent Computation, James P. Crutchfield and Melanie Mitchell (SFI Technical Report 94-03-012)

[26] http://www.santafe.edu/research/topics-information-processing-computation.php#4

[27] The Evolutionary Design of Collective Computation in Cellular Automata, James P. Crutchfeld, Melanie Mitchell, Rajarshi Das (In J. P. Crutch¯eld and P. K. Schuster (editors), Evolutionary Dynamics|Exploring the Interplay of Selection, Neutrality, Accident, and Function. New York: Oxford University Press, 2002.)

[28] Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work, Melanie Mitchell, James P. Crutchfeld, Rajarshi Das (In Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96). Moscow, Russia: Russian Academy of Sciences, 1996.)

[29] Murray, J.. *Mathematical Biology II*. Springer.

[30] http://www.cs.utk.edu/~mclennan/contin-comp.html

[31] Pivato, M: "RealLife: The continuum limit of Larger than Life cellular automata", *Theoretical Computer Science*, 372 (1), March 2007, pp.46-68

[32] Peak, West; Messinger, Mott (2004). "Evidence for complex, collective dynamics and emergent, distributed computation in plants" (http://www.pnas.org/cgi/content/abstract/101/4/918). *Proceedings of the National Institute of Science of the USA* **101** (4): 918–922. Bibcode 2004PNAS..101..918P. doi:10.1073/pnas.0307811100. .

[33] http://gilly.stanford.edu/past_research_files/APackardneuralnet.pdf

[34] Yves Bouligand (1986). *Disordered Systems and Biological Organization*. pp. 374–375.

[35] A. K. Dewdney, The hodgepodge machine makes waves, Scientific American, p. 104, August 1988.

[36] M. Gerhardt and H. Schuster, A cellular automaton describing the formation of spatially ordered structures in chemical systems, Physica D 36, 209-221, 1989.

[37] A. Ilachinsky, Cellular Automata, World Scientific Publishing, 2001, pp. 660.

[38] A. Ilachinsky, Cellular Automata, World Scientific Publishing, 2001, pp. 661-662.

[39] J. P. Crutchfield, "The Calculi of Emergence: Computation, Dynamics, and Induction", Physica D 75, 11-54, 1994.

[40] M. Minsky, "Cellular Vacuum", Int. Jour. of Theo. Phy. 21, 537-551, 1982.

[41] K. Zuse, "The Computing Universe", Int. Jour. of Theo. Phy. 21, 589-600, 1982.

[42] E. Fredkin, "Digital mechanics: an informational process based on reversible universal cellular automata", Physica D 45, 254-270, 1990

[43] iLabs (http://www.ilabs.it/)

[44] F. Berto, G. Rossi, J. Tagliabue, The Mathematics of the Models of Reference, College Publications, 2010 (http://www.mmdr.it/defaultEN.asp)

[45] Weisstein, Eric W.. "Cellular Automaton" (http://mathworld.wolfram.com/CellularAutomaton.html). . Retrieved 13 March 2011.

[46] the Hacker Emblem page on Eric S. Raymond's site (http://www.catb.org/hacker-emblem/)

[47] Blackford, Russell; Ikin, Van; McMullen, Sean (1999), "Greg Egan", *Strange constellations: a history of Australian science fiction*, Contributions to the study of science fiction and fantasy, **80**, Greenwood Publishing Group, pp. 190–200, ISBN 9780313251122; Hayles, N. Katherine (2005), "Subjective cosmology and the regime of computation: intermediation in Greg Egan's fiction", *My mother was a computer: digital subjects and literary texts*, University of Chicago Press, pp. 214–240, ISBN 9780226321479.

[48] Kasman, Alex. "MathFiction: Bloom" (http://kasmana.people.cofc.edu/MATHFICT/mfview.php?callnumber=mf615). . Retrieved 27 March 2011.

[49] http://www.sfwriter.com/syw1.htm

References

- "History of Cellular Automata" (http://www.wolframscience.com/reference/notes/876b) from Stephen Wolfram's *A New Kind of Science*
- Cellular Automata: A Discrete View of the World, Joel L. Schiff, Wiley & Sons, Inc., ISBN 047016879X (0-470-16879-X)
- Chopard, B and Droz, M, 1998, *Cellular Automata Modeling of Physical Systems*, Cambridge University Press, ISBN 0-521-46168-5
- Cellular automaton FAQ (http://cafaq.com/) from the newsgroup comp.theory.cell-automata
- A. D. Wissner-Gross. 2007. *Pattern formation without favored local interactions* (http://www.alexwg.org/publications/JCellAuto_4-27.pdf), Journal of Cellular Automata 4, 27-36 (2008).
- Neighbourhood survey (http://cell-auto.com/neighbourhood/index.html) includes discussion on triangular grids, and larger neighbourhood CAs.
- von Neumann, John, 1966, *The Theory of Self-reproducing Automata*, A. Burks, ed., Univ. of Illinois Press, Urbana, IL.
- Cosma Shalizi's Cellular Automata Notebook (http://cscs.umich.edu/~crshalizi/notebooks/cellular-automata.html) contains an extensive list of academic and professional reference material.
- Wolfram's papers on CAs (http://www.stephenwolfram.com/publications/articles/ca/)
- A.M. Turing. 1952. The Chemical Basis of Morphogenesis. *Phil. Trans. Royal Society*, vol. B237, pp. 37 – 72. (proposes reaction-diffusion, a type of continuous automaton).
- Jim Giles. 2002. What kind of science is this? *Nature* 417, 216 – 218. (discusses the court order that suppressed publication of the rule 110 proof).
- Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work, Melanie Mitchell, James P. Crutchfeld, Rajarshi Das (In Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96). Moscow, Russia: Russian Academy of Sciences, 1996.)
- The Evolutionary Design of Collective Computation in Cellular Automata, James P. Crutchfeld, Melanie Mitchell, Rajarshi Das (In J. P. Crutch¯eld and P. K. Schuster (editors), Evolutionary Dynamics|Exploring the Interplay of Selection, Neutrality, Accident, and Function. New York: Oxford University Press, 2002.)
- The Evolution of Emergent Computation, James P. Crutchfield and Melanie Mitchell (SFI Technical Report 94-03-012)
- Ganguly, Sikdar, Deutsch and Chaudhuri "A Survey on Cellular Automata" (http://www.wepapers.com/Papers/16352/files/swf/15001To20000/16352.swf)
- A. Ilachinsky, Cellular Automata, World Scientific Publishing, 2001 (http://www.ilachinski.com/ca_book.htm)

External links

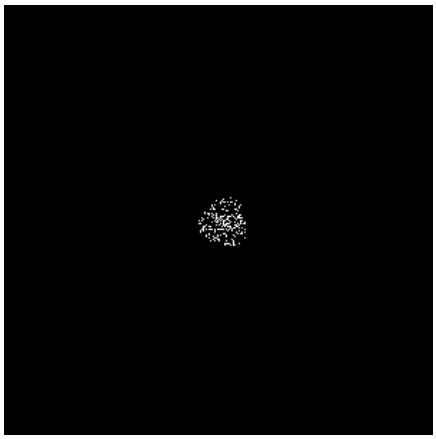
- Mirek's Cellebration (http://www.mirekw.com/ca/index.html) – Home to free MCell and MJCell cellular automata explorer software and rule libraries. The software supports a large number of 1D and 2D rules. The site provides both an extensive rules lexicon and many image galleries loaded with examples of rules. MCell is a Windows application, while MJCell is a Java applet. Source code is available.
- Modern Cellular Automata (http://www.collidoscope.com/modernca/) – Easy to use interactive exhibits of live color 2D cellular automata, powered by Java applet. Included are exhibits of traditional, reversible, hexagonal, multiple step, fractal generating, and pattern generating rules. Thousands of rules are provided for viewing. Free software is available.
- Self-replication loops in Cellular Space (http://necsi.edu/postdocs/sayama/sdsr/java/) – Java applet powered exhibits of self replication loops.

- A collection of over 10 different cellular automata applets (http://vlab.infotech.monash.edu.au/simulations/cellular-automata/) (in Monash University's Virtual Lab)
- Golly (http://www.sourceforge.net/projects/golly) supports von Neumann, Nobili, GOL, and a great many other systems of cellular automata. Developed by Tomas Rokicki and Andrew Trevorrow. This is the only simulator currently available which can demonstrate von Neumann type self-replication.
- Wolfram Atlas (http://atlas.wolfram.com/TOC/TOC_200.html) – An atlas of various types of one-dimensional cellular automata.
- Conway Life (http://www.conwaylife.com/)
- First replicating creature spawned in life simulator (http://www.newscientist.com/article/mg20627653.800-first-replicating-creature-spawned-in-life-simulator.html)
- *The Mathematics of the Models of Reference* (http://www.mmdr.it/provaEN.asp), featuring a general tutorial on CA, interactive applet, free code and resources on CA as model of fundamental physics

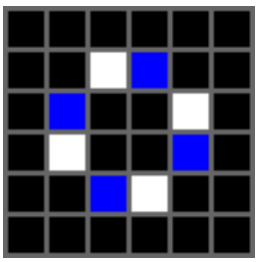
Brian's Brain

Brian's Brain is a cellular automaton devised by Brian Silverman, which is very similar to his Seeds pattern. It consists of an infinite two-dimensional grid of cells, but unlike Seeds, each cell may be in one of three states: on, dying, or off. Each cell is considered to have eight neighbors (Moore neighborhood), as in Seeds and Conway's Game of Life. In each time step, a cell turns on if it was off but had exactly two neighbors that were on, just like the birth rule for Seeds. All cells that were on go into the dying state, which is not counted as an "on" cell in the neighbor count, and prevents any cell from being born there. Cells that were in the dying state go into the off state.

Because of the cellular automaton's name, some websites compare the automaton to a brain and each of its cells to a neuron, which can be in three different states: ready (off), firing (on), and refractory (dying).[1] [2]



A typical chaotic Brian's Brain pattern showing spaceships, rakes and diagonal waves. In this animation, the on cells are white and the dying cells are blue.



An oscillator in Brian's Brain.

The "dying state" cells tend to lead to directional movement, so almost every pattern in Brian's Brain is a spaceship. Many spaceships are rakes, which emit other spaceships. Another result is that many Brian's Brain patterns will explode messily and chaotically, and often will result in or contain great diagonal waves of on and dying cells. For example a 2x2 block of on cells will result in an ever-expanding diamond consisting of four diagonal waves that move across the plane at the pattern's speed of light.

Nevertheless, oscillators have been constructed in Brian's Brain. An example has just four on cells and four dying cells, and oscillates with period 3.[3]

References

- Resnick, Mitchel; Silverman, Brian (1996-02-04). "Exploring Emergence: The Brain Rules" [4] (HTML/Java applet). MIT Media Laboratory, Lifelong Kindergarten Group. Retrieved 2008-12-15.

[1] Evans, M. Steven (2002 -01-28). "Cellular Automata - Brian's Brain" (http://www.msevans.com/automata/briansbrain.html). . Retrieved 2009-05-17.

[2] Wójtowicz, Mirek (2001-09-15). "Cellular Automata rules lexicon - Generations" (http://psoup.math.wisc.edu/mcell/rullex_gene.html#Brian's Brain). *Mirek's Cellebration documentation*. . Retrieved 2009-05-17.

[3] http://midevel.net/forum/viewtopic.php?p=178&sid=dcbc97c2bc13f8a057352434e7b55d0b#187

[4] http://llk.media.mit.edu/projects/emergence/mutants.html

Conway's Game of Life

A single Gosper's Glider Gun creating "gliders"

The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.[1]

The "game" is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

Rules

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, *alive* or *dead*. Every cell interacts with its eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed—births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a *tick* (in other words, each generation is a pure function of the preceding one). The rules continue to be applied repeatedly to create further generations.

Origins

Conway was interested in a problem presented in the 1940s by mathematician John von Neumann, who attempted to find a hypothetical machine that could build copies of itself and succeeded when he found a mathematical model for such a machine with very complicated rules on a rectangular grid. The Game of Life emerged as Conway's successful attempt to drastically simplify von Neumann's ideas. The game made its first public appearance in the October 1970 issue of *Scientific American*, in Martin Gardner's "Mathematical Games" column. From a theoretical point of view, it is interesting because it has the power of a universal Turing machine: that is, anything that can be computed algorithmically can be computed within Conway's Game of Life.[2] [3] Gardner wrote:

> The game made Conway instantly famous, but it also opened up a whole new field of mathematical research, the field of cellular automata ... Because of Life's analogies with the rise, fall and alterations of a society of living organisms, it belongs to a growing class of what are called 'simulation games' (games

that resemble real life processes)

Ever since its publication, Conway's Game of Life has attracted much interest, because of the surprising ways in which the patterns can evolve. Life provides an example of emergence and self-organization. It is interesting for computer scientists, physicists, biologists, biochemists, economists, mathematicians, philosophers, generative scientists and others to observe the way that complex patterns can emerge from the implementation of very simple rules. The game can also serve as a didactic analogy, used to convey the somewhat counter-intuitive notion that "design" and "organization" can spontaneously emerge in the absence of a designer. For example, philosopher and cognitive scientist Daniel Dennett has used the analogue of Conway's Life "universe" extensively to illustrate the possible evolution of complex philosophical constructs, such as consciousness and free will, from the relatively simple set of deterministic physical laws governing our own universe.[4] [5] [6]

The popularity of Conway's Game of Life was helped by its coming into being just in time for a new generation of inexpensive minicomputers which were being released into the market. The game could be run for hours on these machines, which would otherwise have remained unused at night. In this respect, it foreshadowed the later popularity of computer-generated fractals. For many, Life was simply a programming challenge; a fun way to use otherwise wasted CPU cycles. For some, however, Life had more philosophical connotations. It developed a cult following through the 1970s and beyond; current developments have gone so far as to create theoretic emulations of computer systems within the confines of a Life board.[7] [8]

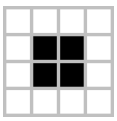
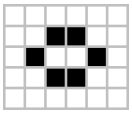
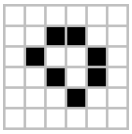
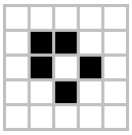
Conway chose his rules carefully, after considerable experimentation, to meet these criteria:

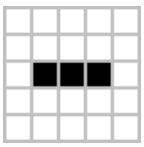
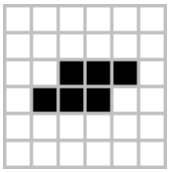
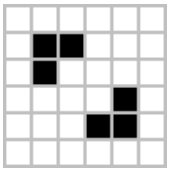
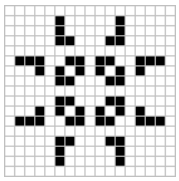
1. There should be no explosive growth, which is why 34 Life was rejected.
2. There should exist small initial patterns with chaotic, unpredictable outcomes.
3. There should be potential for von Neumann universal constructors.
4. The rules should be as simple as possible, whilst adhering to the above constraints.[9]

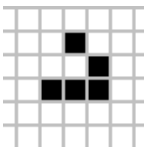
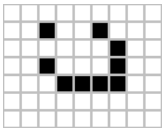
Examples of patterns

The earliest interesting patterns in the Game of Life were discovered without the use of computers. The simplest static patterns ("still lifes") and repeating patterns ("oscillators"—a superset of still lifes) were discovered while tracking the fates of various small starting configurations using graph paper, blackboards, physical game boards (such as Go) and the like. During this early research, Conway discovered that the F-pentomino (which he called the "R-pentomino") failed to stabilize in a small number of generations. In fact, it takes 1103 generations to stabilize, by which time it has a population of 116 and has fired six escaping gliders[10] (in fact, these were the first gliders ever discovered).[11]

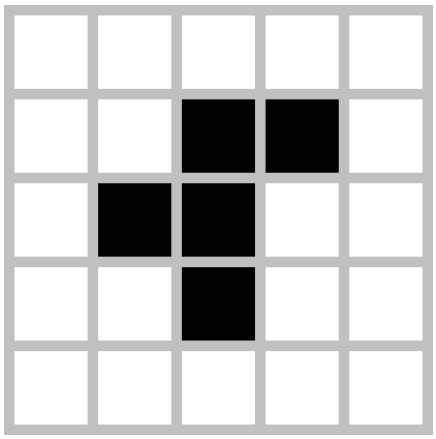
Many different types of patterns occur in the Game of Life, including still lifes, oscillators, and patterns that translate themselves across the board ("spaceships"). Some frequently occurring[12] examples of these three classes are shown below, with live cells shown in black, and dead cells shown in white.

Still lifes	
Block	
Beehive	
Loaf	
Boat	

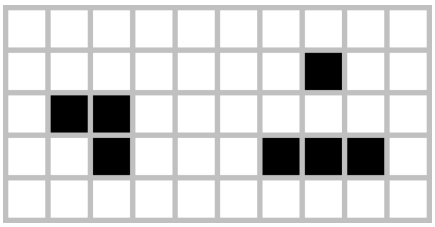
Oscillators	
Blinker (period 2)	
Toad (period 2)	
Beacon (period 2)	
Pulsar (period 3)	

Spaceships	
Glider	
Lightweight spaceship (LWSS)	

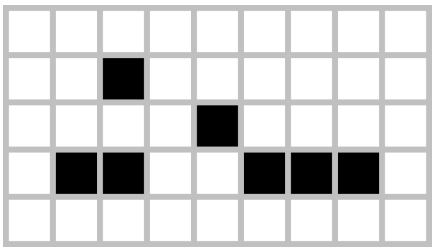
The "pulsar"[13] is the most common period 3 oscillator. The great majority of naturally occurring oscillators are period 2, like the blinker and the toad, but periods 4, 8, 14, 15, 30 and a few others have been seen on rare occasions.[14] Patterns called "Methuselahs" can evolve for long periods before stabilizing, the first-discovered of which was the F-pentomino. "Diehard" is a pattern that eventually disappears (rather than merely stabilize) after 130 generations, which is conjectured to be maximal for patterns with seven or fewer cells.[15] "Acorn" takes 5206 generations to generate 633 cells including 13 escaped gliders.[16]



The F-pentomino



Diehard

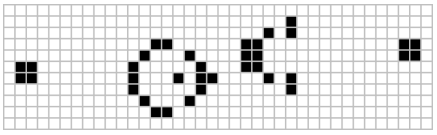


Acorn

Conway originally conjectured that no pattern can grow indefinitely—i.e., that for any initial configuration with a finite number of living cells, the population cannot grow beyond some finite upper limit. In the game's original appearance in "Mathematical Games", Conway offered a $50 prize to the first person who could prove or disprove the conjecture before the end of 1970. One way to disprove it would be to discover patterns that keep adding counters to the field: a "gun", which would be a configuration that repeatedly shoots out moving objects such as the "glider". Another way of disproving the conjecture would be to construct a "puffer train", which would be a

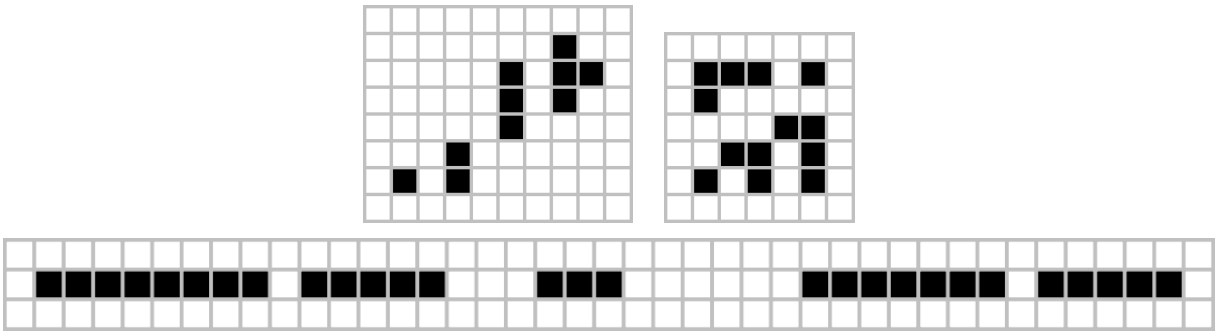
configuration that moves but leaves behind a trail of persistent "smoke".[17]

The prize was won in November of the same year by a team from the Massachusetts Institute of Technology, led by Bill Gosper; the "Gosper glider gun" shown below produces its first glider on the 15th generation, and another glider every 30th generation from then on. This first glider gun is still the smallest one known:[18]



Gosper glider gun

Smaller patterns were later found that also exhibit infinite growth. All three of the following patterns grow indefinitely: the first two create one "block-laying switch engine"[19] each, while the third creates two. The first has only 10 live cells (which has been proven to be minimal).[20] The second fits in a 5 × 5 square. The third is only one cell high:



Later discoveries included other "guns", which are stationary, and which shoot out gliders or other spaceships; "puffers", which move along leaving behind a trail of debris; and "rakes", which move and emit spaceships.[21] Gosper also constructed the first pattern with an asymptotically optimal quadratic growth rate, called a "breeder", or "lobster", which worked by leaving behind a trail of guns.

It is possible for gliders to interact with other objects in interesting ways. For example, if two gliders are shot at a block in just the right way, the block will move closer to the source of the gliders. If three gliders are shot in just the right way, the block will move farther away. This "sliding block memory" can be used to simulate a counter. It is possible to construct logic gates such as *AND*, *OR* and *NOT* using gliders. It is possible to build a pattern that acts like a finite state machine connected to two counters. This has the same computational power as a universal Turing machine, so the Game of Life is theoretically as powerful as any computer with unlimited memory and no time constraints: it is Turing complete.[2] [3]

Furthermore, a pattern can contain a collection of guns that fire gliders in such a way as to construct new objects, including copies of the original pattern. A "universal constructor" can be built which contains a Turing complete computer, and which can build many types of complex objects, including more copies of itself.[22]

Self-replication

On May 18, 2010, Andrew J. Wade announced a self-constructing pattern dubbed Gemini which creates a copy of itself while destroying its parent.[23] [24] This pattern replicates in 34 million generations, and uses an instruction tape made of gliders which oscillate between two stable configurations made of Chapman-Greene construction arms. These, in turn, create new copies of the pattern, and destroy the previous copy. Gemini is a spaceship.[25]

Iteration

From a random initial pattern of living cells on the grid, observers will find the population constantly changing as the generations tick by. The patterns that emerge from the simple rules may be considered a form of beauty. Small isolated subpatterns with no initial symmetry tend to become symmetrical. Once this happens, the symmetry may increase in richness, but it cannot be lost unless a nearby subpattern comes close enough to disturb it. In a very few cases the society eventually dies out, with all living cells vanishing, though this may not happen for a great many generations. Most initial patterns eventually "burn out", producing either stable figures or patterns that oscillate forever between two or more states;[26] [27] many also produce one or more gliders or spaceships that travel indefinitely away from the initial location. Spaceships can travel at most one cell per unit time: so this velocity is set to be the cellular automaton speed of light and denoted c.

Algorithms

Early patterns with unknown futures, such as the R-pentomino, led computer programmers across the world to write programs to track the evolution of Life patterns. Most of the early algorithms were similar; they represented Life patterns as two-dimensional arrays in computer memory. Typically two arrays are used, one to hold the current generation, and one in which to calculate its successor. Often 0 and 1 represent dead and live cells respectively. A nested for-loop considers each element of the current array in turn, counting the live neighbours of each cell to decide whether the corresponding element of the successor array should be 0 or 1. The successor array is displayed. For the next iteration the arrays swap roles so that the successor array in the last iteration becomes the current array in the next iteration.

A variety of minor enhancements to this basic scheme are possible, and there are many ways to save unnecessary computation. A cell that did not change at the last time step, and none of whose neighbours changed, is guaranteed not to change at the current time step as well. So, a program that keeps track of which areas are active can save time by not updating the inactive zones.[28]

Glider gun with toroidal array

In principle, the Life field is infinite, but computers have finite memory, and array sizes must usually be declared in advance. This leads to problems when the active area encroaches on the border of the array. Programmers have used several strategies to address these problems. The simplest strategy is simply to assume that every cell outside the array is dead. This is easy to program, but leads to inaccurate results when the active area crosses the boundary. A more sophisticated trick is to consider the left and right edges of the field to be stitched together, and the top and bottom edges also, yielding a toroidal array. The result is that active areas that move across a field edge reappear at the opposite edge. Inaccuracy can still result if the pattern grows too large, but at least there are no pathological edge effects. Techniques of dynamic storage allocation may also be used, creating ever-larger arrays to hold growing patterns.

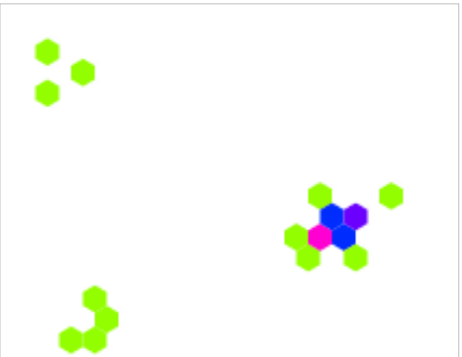
Alternatively, the programmer may abandon the notion of representing the Life field with a 2-dimensional array, and use a different data structure, like a vector of coordinate pairs representing live cells. This approach allows the pattern to move about the field unhindered, as long as the population does not exceed the size of the live-coordinate array. The drawback is that counting live neighbours becomes a search operation, slowing down simulation speed. With more sophisticated data structures this problem can also be largely solved.

For exploring large patterns at great time-depths, sophisticated algorithms such as Hashlife may be useful. There is also a method, applicable to other cellular automata too, for implementation of the Game of Life using arbitrary asynchronous updates whilst still exactly emulating the behaviour of the synchronous game.[29]

Variations on Life

Since Life's inception, new similar cellular automata have been developed. The standard Game of Life, in which a cell is "born" if it has exactly 3 neighbours, stays alive if it has 2 or 3 living neighbours, and dies otherwise, is symbolised as "B3/S23". The first number, or list of numbers, is what is required for a dead cell to be born. The second set is the requirement for a live cell to survive to the next generation. Hence "B6/S16" means "a cell is born if there are 6 neighbours, and lives on if there are either 1 or 6 neighbours". Cellular automata on a two-dimensional grid that can be described in this way are known as Life-like cellular automata. Another common Life-like automaton, *HighLife*, is described by the rule B36/S23, because having 6 neighbours, in addition to the original game's B3/S23 rule, causes a birth. HighLife is best known for its frequently occurring replicators.[30] [31] Additional Life-like cellular automata exist, although the vast majority of them produce universes that are either too chaotic or too desolate to be of interest.

Some variations on Life modify the geometry of the universe as well as the rule. The above variations can be thought of as 2-D square, because the world is two-dimensional and laid out in a square grid. 1-D square variations (known as elementary cellular automata)[32] and 3-D square variations have been developed, as have 2-D hexagonal and 2-D triangular variations.



A sample of a 48-step oscillator along with a 2-step oscillator and a 4-step oscillator from a 2-D hexagonal Game of Life (rule H:B2/S34)

Conway's rules may also be generalized such that instead of two states (*live* and *dead*) there are three or more. State transitions are then determined either by a weighting system or by a table specifying separate transition rules for each state; for example, Mirek's Cellebration's multi-coloured "Rules Table" and "Weighted Life" rule families each include sample rules equivalent to Conway's Life.

Patterns relating to fractals and fractal systems may also be observed in certain Life-like variations. For example, the automaton B1/S12 generates four very close approximations to the Sierpiński triangle when applied to a single live cell. The Sierpiński triangle can also be observed in Conway's Game of Life by examining the long-term growth of a long single-cell-thick line of live cells,[33] as well as in HighLife, Seeds (B2/S), and Wolfram's Rule 90.[34]

Immigration is a variation that is very similar to Conway's Game of Life, except that there are two ON states (often expressed as two different colours). Whenever a new cell is born, it takes on the ON state that is the majority in the three cells that gave it birth. This feature can be used to examine interactions between spaceships and other "objects" within the game.[35] Another similar variation, called *QuadLife*, involves four different ON states. When a new cell is born from three different ON neighbours, it takes on the fourth value, and otherwise, like Immigration, it takes the majority value.[36] Except for the variation among ON cells, both of these variations act identically to Life.

Entertainment

Video Games

Some games for entertainment purposes have been developed from the Game of Life. One such game, for two players who each interact with the "game" once per tick, is based directly upon Conway's Game of Life. Live cells have one of two colours, and a player wins when all cells of the opponent's colour are eliminated. When a dead cell has three live neighbours, it becomes live in the same colour as the majority of its neighbours, as in the aforementioned *Immigration*. The game starts with a random or pre-chosen starting pattern with half the live cells of each colour. After one iteration, the first player may add one cell of his or her own colour and remove one cell of his or her opponent's colour. After the next iteration, the other player can do the same, and so forth. Other two-player variants of the Game of Life have also been developed.[37]

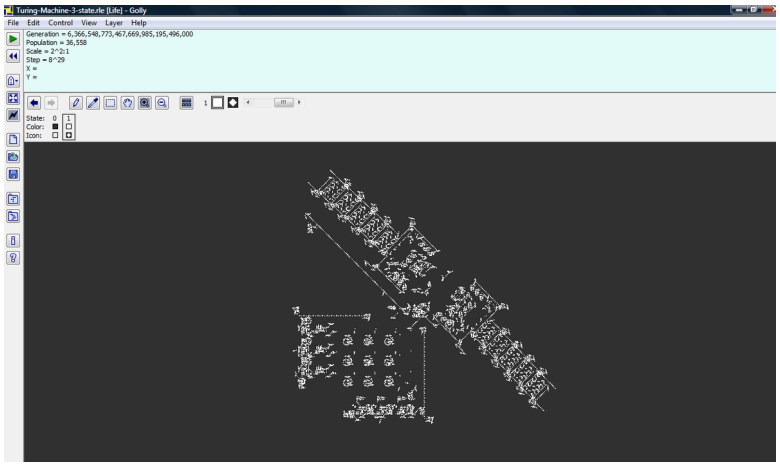
In Populous II, one of the 'divine intervention' effects is a fungus that grows according to the rules of the Game of Life.

Music

Various musical composition techniques use Conway's Game of Life, especially in MIDI sequencing.[38] A variety of programs exist for creating sound from patterns generated in the Game of Life. (see footnotes for links to examples.)[39] [40] [41]

The Native Instruments modular sound generation/processing software package Reaktor features a drum machine with an integrated sequencer which implements the Game of Life.

Notable Life programs



The 6,366,548,773,467,669,985,195,496,000th (6 octillionth) generation of a Turing machine, made in the game of Life, computed in less than 30 seconds on an Intel Duo 2GHz CPU using Golly in Hashlife mode

Computers have been used to follow Life configurations from the earliest days. When John Conway was first investigating how various starting configurations developed, he tracked them by hand using a Go board with its black and white stones. This was tedious, and prone to errors. The first-ever Life program was written by John Francis (an undergraduate student at Cambridge) on an IBM 360, and was used to automate this process and track the fate of the "R" pentomino for 1000 generations. The first interactive Life program was written in ALGOL 68 for the PDP-7 by M. J. T. Guy and S. R. Bourne. Without its help, some discoveries about the game would have been difficult to make. The results were published in the October 1970 issue of Scientific American.[42]

There are now thousands of Life programs on line, so a full list will not be provided here. The following is a selection of a small number of programs with some special claim to notability, such as popularity or unusual features. Most of these programs incorporate a graphical user interface for pattern editing and simulation, the capability for simulating multiple rules including Life, and a large library of interesting patterns in Life and other CA rules.

- Conway's Game of Life [43], by Alan Hensel. A pop-up Java web applet with fast simulation algorithms and a large library of interesting Life patterns.
- Golly [44]. A cross-platform (Windows, Macintosh and Linux) open-source simulation system for Life and other cellular automata, by Andrew Trevorrow and Tomas Rokicki. It includes the hashlife algorithm for extremely fast generation, and Perl or Python scriptability for both editing and simulation.
- Life32 [45]. Freeware for Windows machines, it includes powerful and scriptable pattern editing features.
- Mirek's Cellebration [46]. Free 1-D and 2-D cellular automata viewer, explorer and editor for Windows. Includes powerful facilities for simulating and viewing a wide variety of CA rules including Life, and a scriptable editor.
- Xlife [47]. A cellular-automaton laboratory by Jon Bennett. The standard Linux Life simulation application for a long time, it has also been ported to Windows. Can handle cellular automaton rules with the same neighbourhood as Life, and up to eight possible states per cell.

References

[1] Gardner, Martin (1970-10). *Mathematical Games - The fantastic combinations of John Conway's new solitaire game "life"* (http://web.archive.org/web/20090603015231/http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm). **223**. pp. 120–123. ISBN 0894540017. Archived from the original (http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm) on 2009-06-03. . Retrieved 2011-06-26.

[2] Paul Chapman (November 11, 2002). "Life Universal Computer" (http://www.igblan.free-online.co.uk/igblan/ca/). . Retrieved July 12, 2009.

[3] Berlekamp, E. R.; ; Guy, R.K. (2001 2004). *Winning Ways for your Mathematical Plays* (2nd ed.). A K Peters Ltd. ISBN 978-1-56881-130-7; ISBN 156881142X; ISBN 1568811438; ISBN 1568811446

[4] Dennett, D.C. (1991). *Consciousness Explained*. Boston: Back Bay Books. ISBN 0316180661

[5] Dennett, D.C. (1995). *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. New York: Simon & Schuster. ISBN 068482471X

[6] Dennett, D.C. (2003). *Freedom Evolves*. New York: Penguin Books. ISBN 0142003840

[7] Paul Rendell (January 12, 2005). "A Turing Machine in Conway's Game of Life" (http://rendell-attic.org/gol/tm.htm). . Retrieved July 12, 2009.

[8] Adam P. Goucher. "Spartan universal computer-constructor" (http://www.conwaylife.com/wiki/index.php?title=Spartan_universal_computer-constructor). LifeWiki. . Retrieved July 12, 2009.

[9] Conway, private communication to the 'Life list', 14 April 1999.

[10] "R-pentomino" (http://www.conwaylife.com/wiki/index.php?title=R-pentomino). LifeWiki. . Retrieved July 12, 2009.

[11] Stephen A. Silver. "Glider" (http://www.argentum.freeserve.co.uk/lex_g.htm#glider). The Life Lexicon. . Retrieved July 12, 2009.

[12] "Census Results in Conway's Game of Life" (http://www.conwaylife.com/soup/census.asp?rule=B3/S23&sl=1&os=1&ss=1). The Online Life-Like CA Soup Search. . Retrieved July 12, 2009.

[13] "Pulsar" (http://www.ericweisstein.com/encyclopedias/life/Pulsar.html). Eric Weisstein's Treasure Trove of Life. . Retrieved 2008-09-16.

[14] Achim Flammenkamp (2004-09-07). "Most seen natural occurring ash objects in Game of Life" (http://wwwhomes.uni-bielefeld.de/achim/freq_top_life.html). . Retrieved 2008-09-16.

[15] Stephen A. Silver. "Diehard" (http://www.argentum.freeserve.co.uk/lex_d.htm#diehard). The Life Lexicon. . Retrieved July 12, 2009.

[16] Koenig, H. (February 21, 2005). "New Methuselah Records" (http://pentadecathlon.com/lifeNews/2005/02/new_methuselah_records.html). . Retrieved January 24, 2009.

[17] Stephen A. Silver. "Puffer train" (http://www.argentum.freeserve.co.uk/lex_p.htm#puffertrain). The Life Lexicon. . Retrieved July 12, 2009.

[18] Stephen A. Silver. "Gosper glider gun" (http://www.argentum.freeserve.co.uk/lex_g.htm#gosperglidergun). The Life Lexicon. . Retrieved July 12, 2009.

[19] "Block-laying switch engine" (http://www.conwaylife.com/wiki/index.php?title=Block-laying_switch_engine). LifeWiki. . Retrieved July 12, 2009.

[20] "Infinite Growth" (http://www.ericweisstein.com/encyclopedias/life/InfiniteGrowth.html). Eric Weisstein's Treasure Trove of Life. . Retrieved 2008-09-16.

[21] Stephen A. Silver. "Rake" (http://www.argentum.freeserve.co.uk/lex_r.htm#rake). The Life Lexicon. . Retrieved July 12, 2009.

[22] Descriptions of these constructions are given in Berlekamp, E. R.; ; Guy, R.K. (2001 2004). *Winning Ways for your Mathematical Plays* (2nd ed.). A K Peters Ltd. ISBN 978-1-56881-130-7; ISBN 156881142X; ISBN 1568811438; ISBN 1568811446

[23] http://conwaylife.com/forums/viewtopic.php?f=2&t=399&p=2327#p2327

[24] http://conwaylife.com/wiki/index.php?title=Gemini

[25] Aron, Jacob (16 June 2010). "First replicating creature spawned in life simulator". *New Scientist*

[26] Andrzej Okrasinski. "Game of Life Object Statistics" (http://web.archive.org/web/20090727010353/http://geocities.com/conwaylife/). Archived from the original (http://www.geocities.com/conwaylife/) on 2009-07-27. . Retrieved July 12, 2009.
[27] Nathaniel Johnston. "The Online Life-Like CA Soup Search" (http://www.conwaylife.com/soup/). . Retrieved July 12, 2009.
[28] Alan Hensel. "About my Conway's Game of Life Applet" (http://www.ibiblio.org/lifepatterns/lifeapplet.html). . Retrieved July 12, 2009.
[29] Nehaniv, Chrystopher L. (2002). "Self-Reproduction in Asynchronous Cellular Automata". *2002 NASA/DoD Conference on Evolvable Hardware (15–18 July 2002, Alexandria, Virginia, USA)*. IEEE Computer Society Press. pp. 201–209
[30] HighLife - An Interesting Variant of Life (http://www.tip.net.au/~dbell/articles/HighLife.zip) by David Bell (.zip file)
[31] Stephen A. Silver. "Replicator" (http://www.argentum.freeserve.co.uk/lex_r.htm#replicator). The Life Lexicon. . Retrieved July 12, 2009.
[32] "Elementary Cellular Automaton" (http://mathworld.wolfram.com/ElementaryCellularAutomaton.html). Wolfram Mathworld. . Retrieved July 12, 2009.
[33] "One cell thick pattern" (http://www.conwaylife.com/wiki/index.php?title=One_cell_thick_pattern). LifeWiki. . Retrieved July 12, 2009.
[34] "Life Imitates Sierpinski" (http://www.conwaylife.com/forums/viewtopic.php?f=7&t=90). ConwayLife.com forums. . Retrieved July 12, 2009.
[35] "Immigration" (http://www.ericweisstein.com/encyclopedias/life/Immigration.html). Eric Weisstein's Treasure Trove of Life. . Retrieved 2008-09-16.
[36] "QuadLife" (http://www.ericweisstein.com/encyclopedias/life/QuadLife.html). Eric Weisstein's Treasure Trove of Life. . Retrieved 2008-09-16.
[37] Mark Levene; George Roussos (2002). "A Two-Player Game of Life". *International Journal of Modern Physics C [Computational Physics and Physical Computation]* **14** (2): 195. arXiv:cond-mat/0207679. doi:10.1142/S0129183103004346.
[38] Burraston, Dave; Edmonds, Ernest (2004). "Cellular Automata in MIDI based Computer Music" (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.3882&rep=rep1&type=pdf). *Proceedings of the 2004 International Computer Music Conference*. doi:10.1.1.6.3882.
[39] glitchDS – Cellular Automaton Sequencer For The Nintendo DS (http://www.synthtopia.com/content/2008/05/29/glitchds-cellular-automaton-sequencer-for-the-nintendo-ds/)
[40] Game Of Life Music Sequencer (http://www.synthtopia.com/content/2009/04/29/game-of-life-music-sequencer/)
[41] Game Of Life Music Sequencer For iOS, Runxt Life (http://www.synthtopia.com/content/2011/01/12/game-of-life-music-sequencer-for-ios-runxt-life/)
[42] Gardner, Martin (October 1970). "Mathematical Games: The fantastic combinations of John Conway's new solitaire game "Life"". *Scientific American* **223**: 120–123
[43] http://www.ibiblio.org/lifepatterns/
[44] http://golly.sourceforge.net/
[45] http://www.brothersoft.com/life32-download-298213.html
[46] http://www.mirekw.com/ca/index.html
[47] http://packages.debian.org/lenny/xlife

External links

- Conway's Game of Life (http://www.dmoz.org/Computers/Artificial_Life/Cellular_Automata/Conway's_Game_of_Life/) at the Open Directory Project
- Game of Life News (http://pentadecathlon.com/lifeNews/index.php)
- LifeWiki (http://www.conwaylife.com/wiki/)
- Cellular Automata FAQ — Conway's Game of Life (http://cafaq.com/lifefaq/index.php)

Langton's ant

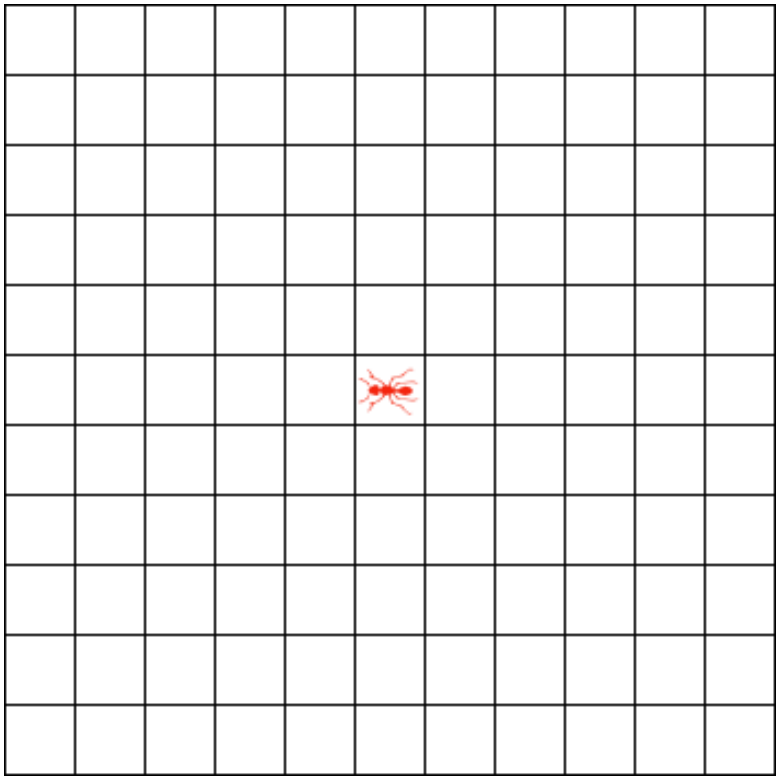
Langton's ant is a two-dimensional Turing machine with a very simple set of rules but complicated emergent behavior. It was invented by Chris Langton in 1986 and runs on a square lattice of black and white cells.[1] The universality of Langton's ant was proven in 2000.[2] The idea has been generalized in several different ways, such as turmites which add more colors and more states.

Langton's ant after 11000 steps. A red pixel shows the ant's location.

Rules

Squares on a plane are colored variously either black or white. We arbitrarily identify one square as the "ant". The ant can travel in any of the four cardinal directions at each step it takes. The ant moves according to the rules below:

- At a white square, turn 90° right, flip the color of the square, move forward one unit
- At a black square, turn 90° left, flip the color of the square, move forward one unit



Animation of first 200 steps of Langton's ant

These simple rules lead to surprisingly complex behavior: after an initial period of apparently chaotic behavior, that lasts for about 10,000 steps (in the simplest case), the ant starts building a recurrent "highway" pattern of 104 steps that repeat indefinitely. All finite initial configurations tested eventually converge to the same repetitive pattern suggesting that the "highway" is an attractor of Langton's ant, but no one has been able to prove that this is true for all such initial configurations. It is only known that the ant's trajectory is always unbounded regardless of the initial configuration[3] - this is known as the **Cohen-Kung theorem**.[4]

Langton's ant can also be described as a cellular automaton, where most of the grid is colored black or white, and the "ant" square has one of eight different colors assigned to encode the combination of black/white state and the current direction of motion of the ant.

Universality

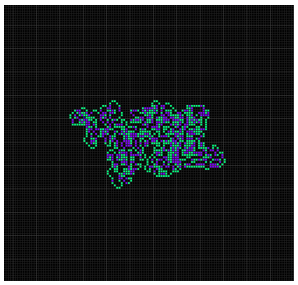
In 2000, Gajardo et al. showed a construction that calculates any boolean circuit using the trajectory of a single instance of Langton's ant.[2] Thus, it would be possible to simulate a Turing machine using the ant's trajectory for computation. This means that the ant is capable of universal computation.

Extension to multiple colors

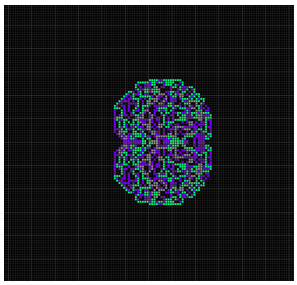
Greg Turk and Jim Propp considered a simple extension to Langton's ant where instead of just two colors, more colors are used.[5] The colors are modified in a cyclic fashion. A simple naming scheme is used: for each of the successive colors, a letter 'L' or 'R' is used to indicate whether a left or right turn should be taken. Langton's ant has the name 'RL' in this naming scheme.

Some of these extended Langton's ants produce patterns that become symmetric over and over again. One of the simplest examples is the ant 'RLLR'. One sufficient condition for this to happen is that the ant's name, seen as a cyclic list, consists of consecutive pairs of identical letters 'LL' or 'RR' (the term "cyclic list" indicates that the last letter may pair with the first one.) The proof involves Truchet tiles.

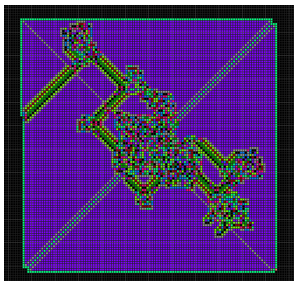
Some example patterns in the multiple-color extension of Langton's Ants:



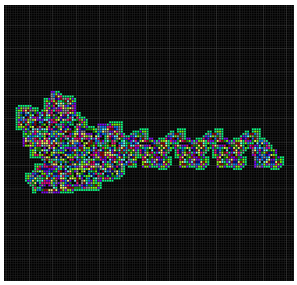
RLR: grows chaotically. It is not known if this ant ever produces a highway.



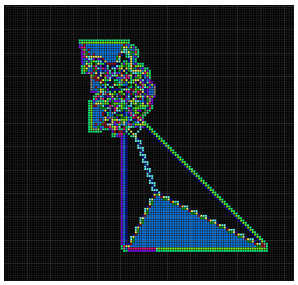
LLRR: grows symmetrically.



LRRRRRLLR: fills space in a square around itself.



LLRRRLRLRLLR: creates a convoluted highway.

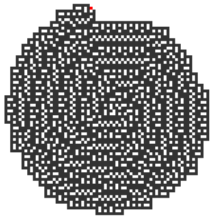


RRLLLRLLLRRR: creates a filled triangle shape that grows and moves.

Extension to multiple states

A further extension of Langton's Ants is to consider multiple states of the Turing machine - as if the ant itself has a color that can change. These ants are called turmites, a contraction of "Turing machine termites". Common behaviours include the production of highways, chaotic growth and spiral growth.[6]

Some example turmites:



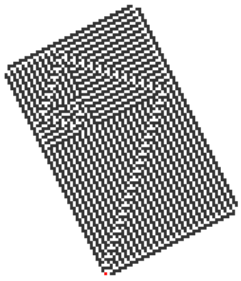
Spiral growth.

Semi-chaotic growth.

Production of a highway after a period of chaotic growth.

Chaotic growth with a distinctive texture.

Growth with a distinctive texture inside an expanding frame.



Constructing a Fibonacci spiral.

Extension to multiple ants

Multiple Langton's ants can co-exist on the 2D plane, as long as there is a rule for what happens when they meet. Ed Pegg, Jr. considered turmites that can turn for example *both* left and right, splitting in two and annihilating each other when they meet.[7]

References

[1] Langton, Chris G. (1986). "Studying artificial life with cellular automata". *Physica D: Nonlinear Phenomena* **22** (1-3): 120–149. doi:10.1016/0167-2789(86)90237-X. hdl:2027.42/26022.

[2] Gajardo, A.; A. Moreira, E. Goles (15 March 2002). "Complexity of Langton's ant" (http://www.dim.uchile.cl/~anmoreir/oficial/langton_dam.pdf). *Discrete Applied Mathematics* **117** (1-3): 41–50. doi:10.1016/S0166-218X(00)00334-6. .

[3] Bunimovich, Leonid A.; Serge E. Troubetzkoy (1992). "Recurrence properties of Lorentz lattice gas cellular automata". *Journal of Statistical Physics* **67** (1-2): 289–302. doi:10.1007/BF01049035.

[4] Weisstein, Eric W., " Cohen-Kung Theorem (http://mathworld.wolfram.com/Cohen-KungTheorem.html)" from MathWorld.

[5] Gale, D.; J. Propp, S. Sutherland, S.Troubetzkoy (1995). "Further Travels with My Ant" (http://www.math.sunysb.edu/cgi-bin/preprint.pl?ims95-1). *Mathematical Entertainments column, Mathematical Intelligencer* **17**: 48–56. .

[6] Pegg, Jr., Ed. *Turmite* (http://mathworld.wolfram.com/Turmite.html). From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein. . Retrieved 15 October 2009.

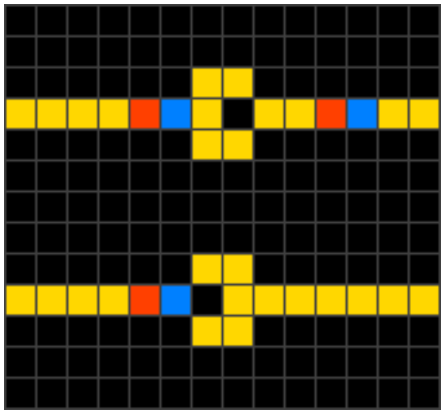
[7] Pegg, Jr., Ed. "Math Puzzle" (http://www.mathpuzzle.com/26Mar03.html). . Retrieved 15 October 2009.

External links

- Weisstein, Eric W., " Langton's Ant (http://mathworld.wolfram.com/LangtonsAnt.html)" from MathWorld.
- Online demonstration of Langton's ant (http://www.math.ubc.ca/~cass/www/ant/ant.html)
- Generalized Ants (http://www.math.sunysb.edu/~scott/ants/)
- JavaScript Demonstration (http://skyward.nefec.org/langstonsAnt.php)
- Another JavaScript Demonstration (http://wendlinger.org/jowe/langton_ant.html)
- Java applet with multiple colours and programmable ants (http://www.hut.fi/~jblomqvi/langton/index.html)
- Langton's ant in 3-D (examples and small demo program) (http://heikohamann.de/langtonsAnt/langtons_ant_3d.html)
- Mathematical Recreations column (http://www.fortunecity.com/emachines/e11/86/langton.html) by Ian Stewart using Langton's Ant as a metaphor for a Theory of everything. Contains the proof that Langton's ant is unbounded.
- Java applet on several grids and editable graphs, it shows how the ant can compute logical gates (http://www.ing-mat.udec.cl/~anahi/langton)
- Programming Langton's ants (http://www.willmcgugan.com/2007/05/18/langton-ants-in-pygame/) in Python using Pygame.
- A video demo of different multiple-color Langton's Ants (http://www.youtube.com/watch?v=1X-gtr4pEBU)
- Golly script for generating rules in the multiple color extension of Langton's ant (http://ruletablerepository.googlecode.com/files/Langtons-Ant-nColor.zip)

Wireworld

Wireworld is a cellular automaton first proposed by Brian Silverman in 1987, as part of his program Phantom Fish Tank. It subsequently became more widely known as a result of an article in the "Computer Recreations" column of *Scientific American*.[1] Wireworld is particularly suited to simulating electronic logic elements, or "gates", and, despite the simplicity of the rules, Wireworld is Turing-complete.



2 Wireworld diodes, the above one in conduction direction, the lower one in reverse-biasing

Rules

A Wireworld cell can be in one of four different states:

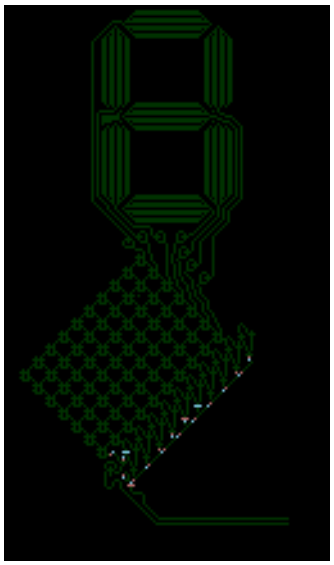
1. Empty
2. Electron head
3. Electron tail
4. Conductor

Software often numbers the states 0-3 rather than 1-4. In the examples given here, the states are displayed arbitrarily as colours proceeding through: black, blue, red, yellow.

Like in all cellular automata, time proceeds in discrete steps called generations (sometimes "gens" or "ticks"). Cells behave as follows:

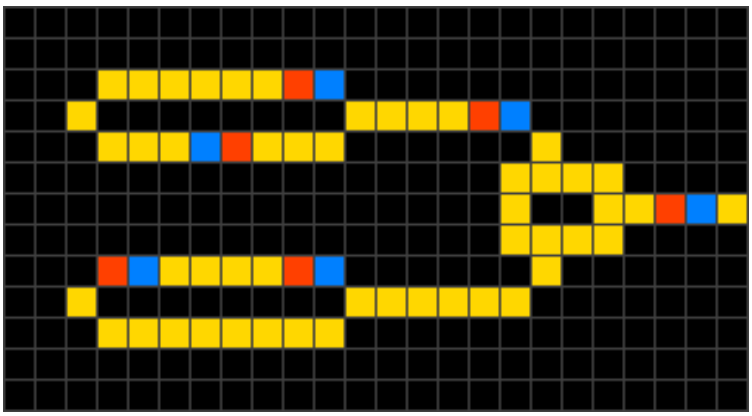
- Empty → Empty
- Electron head → Electron tail
- Electron tail → Conductor
- Conductor → Electron head if exactly one or two of the neighbouring cells are electron heads, or remains Conductor otherwise.

Wireworld uses what is called the Moore neighborhood, which means that in the rules above, neighbouring means one cell away (range value of one) in any direction, both orthogonal and diagonal.



Example of a complicated circuit made in WireWorld: a seven segment display and decoder. Conductor cells are dark green to highlight signal flow and display segments.

These simple rules can be used to construct logic gates (see below).



2 Clock generators sending electrons into an XOR gate

Applications

Entities built within Wireworld universes include a computer which enumerates prime numbers[2] , and unit cells for another cellular automaton; Langton's Ant (allowing any Langton's Ant pattern to be built within Wireworld).[3]

Computer programs featuring Wireworld

- Wireworlds [4]: the classical Wireworld cellular automaton and many variations. Game contains about 100 examples of logical gates etc created in Wireworld.
- A Wireworld Computer [5]: an account of the construction of a full-scale computer using wireworld.
- MCell [6]: A free program for running cellular automata, can run Wireworlds.
- Golly [7]: Probably the fastest way to run large Wireworld configurations. The Wireworld computer is supplied in the examples.
- Wireworld player [8]: A Flash-based Wireworld simulator, with support for plain-text and MCell formats.
- Wiresq [9]: a music synthesizer for iOS devices.
- The Powder Toy [10]: a computer based sandbox particle simulation program, featured in v55.1+.

References

[1] Computer recreations: The cellular automata programs that create Wireworld, Rugworld and other diversions, Scientific American (1990) by A K Dewdney
[2] Mark Owen. "The Wireworld Computer" (http://www.quinapalus.com/wi-index.html). .
[3] Nyles Heise. "Wireworld" (http://www.heise.ws/wireworld.html). .
[4] http://www.zillions-of-games.com/cgi-bin/zilligames/submissions.cgi/5295?do=show;id=2
[5] http://www.quinapalus.com/wi-index.html
[6] http://mirekw.com/ca/index.html
[7] http://sourceforge.net/projects/golly/
[8] http://www.rezmason.net/wireworld/index.html
[9] http://heavyephemera.com/wiresq/
[10] http://powdertoy.co.uk/Download.html

External links

- Wireworld (http://rosettacode.org/wiki/Wireworld) on Rosetta Code
- The Wireworld computer in Java (http://www.quinapalus.com/wi-java.html)
- wireworld GUI in python (http://perso.ens-lyon.fr/martin.bodin/L3/projet/2/index.html)

Elementary cellular automaton

In mathematics and computability theory, an **elementary cellular automaton** is a one-dimensional cellular automaton where there are two possible states (labeled 0 and 1) and the rule to determine the state of a cell in the next generation depends only on the current state of the cell and its two immediate neighbors. As such it is one of the simplest possible models of computation. Nevertheless, there is an elementary cellular automaton (rule 110, defined below) which is capable of universal computation.

The numbering system

There are $8 = 2^3$ possible configurations for a cell and its two immediate neighbors. The rule defining the cellular automaton must specify the resulting state for each of these possibilities so there are $256 = 2^8$ possible elementary cellular automata. Stephen Wolfram proposed a scheme, known as the Wolfram code, to assign each rule a number from 0 to 255 which has become standard. Each possible current configuration is written on order, 111, 110, ... , 001, 000, and the resulting state for each of these configurations is written in the same order and interpreted as the binary representation of an integer. This number is taken to be the rule number of the automaton. For example, 110 written in binary is 01101110_2. So rule 110 is defined by the transition rule:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Reflections and complements

Although there are 256 possible rules, many of these are trivially equivalent to each other up to a simple transformation of the underlying geometry. The first such transformation is reflection through a vertical axis and the result of applying this transformation to a given rule is called the **mirrored rule**. These rules will exhibit the same behavior up to reflection through a vertical axis, and so are equivalent in a computational sense.

For example, if the definition of rule 110 is reflected through a vertical line, the following rule (rule 124) is obtained:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	1	1	1	0	0

Rules which are the same as their mirrored rule are called **amphichiral**. Of the 256 elementary cellular automata, 64 are amphichiral.

The second such transformation is to exchange the roles of 0 and 1 in the definition. The result of applying this transformation to a given rule is called the **complementary rule**. For example, if this transformation is applied to rule 110, the following rule (rule 137) obtained:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	1	0	0	0	1	0	0	1

There are 16 rules which are the same as their complementary rules.

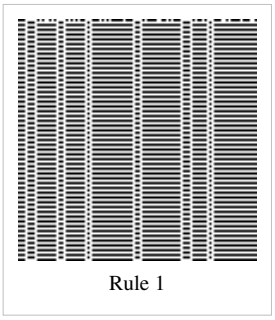
Finally, the previous two transformations can be applied successively to a rule to obtain the mirrored complementary rule. For example, the mirrored complementary rule of rule 110 is rule 193. There are 16 rules which are the same as their mirrored complementary rules.

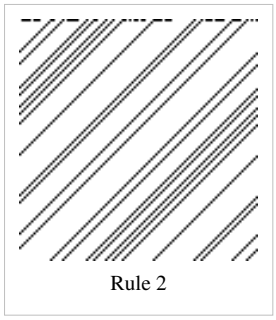
Of the 256 elementary cellular automata, there are 88 which are inequivalent under these transformations.

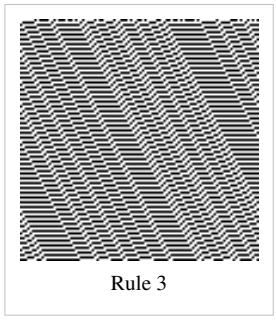
Unique rules

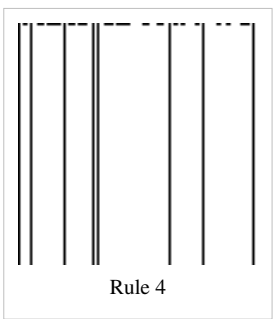
The 88 inequivalent rules are as follows, evolved from random initial conditions:

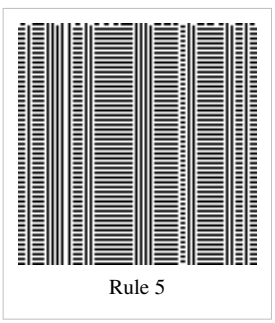

Rule 0

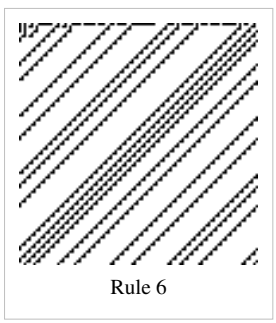

Rule 1

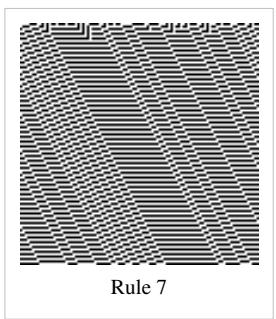

Rule 2


Rule 3

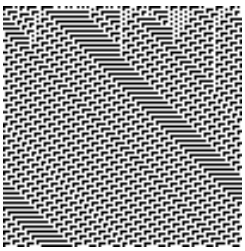

Rule 4

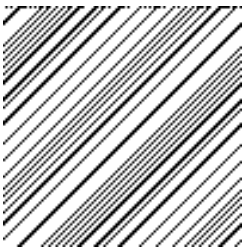

Rule 5

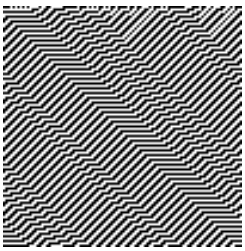

Rule 6

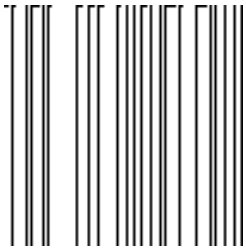

Rule 7

Rule 8

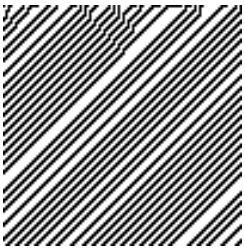

Rule 9

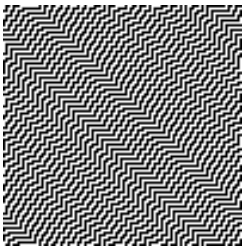

Rule 10

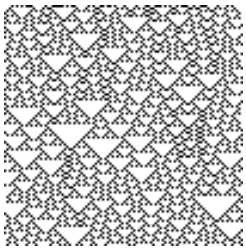

Rule 11

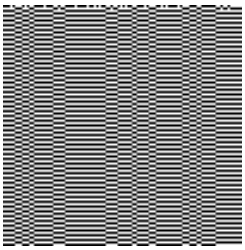

Rule 12

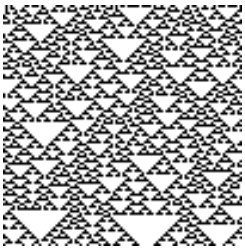

Rule 13

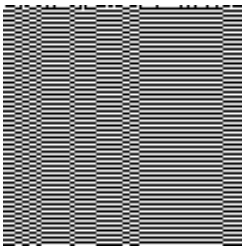

Rule 14

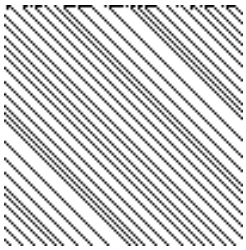

Rule 15

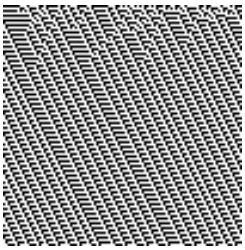

Rule 18

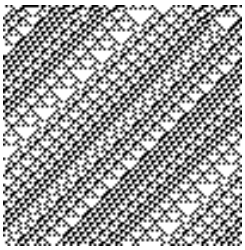

Rule 19

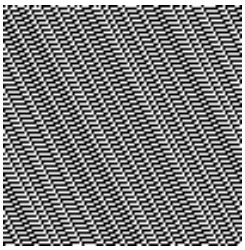

Rule 22

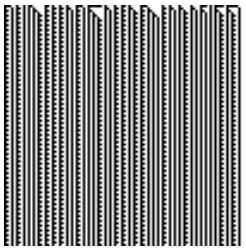

Rule 23

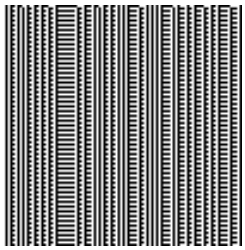

Rule 24


Rule 25


Rule 26

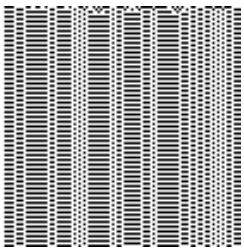

Rule 27


Rule 28


Rule 29

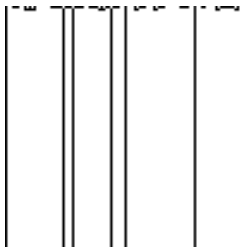

Rule 30

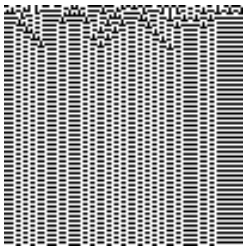
Rule 32


Rule 33


Rule 34

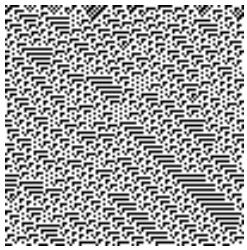

Rule 35


Rule 36

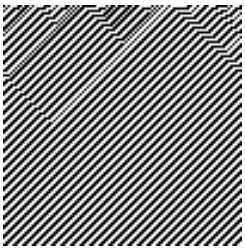

Rule 37

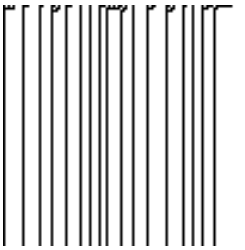

Rule 38

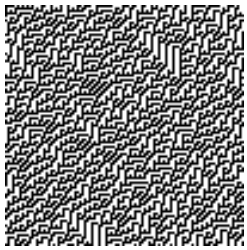
Rule 40


Rule 41

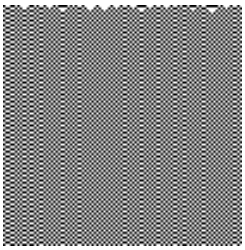

Rule 42

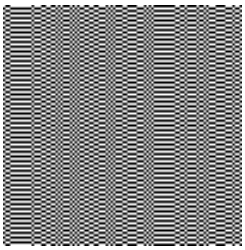

Rule 43

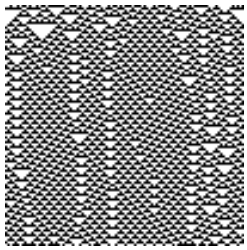

Rule 44

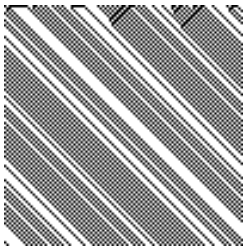

Rule 45

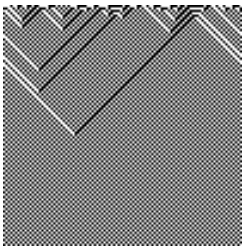

Rule 46


Rule 50

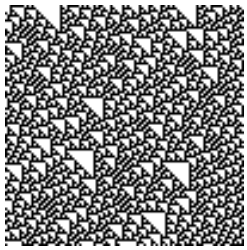

Rule 51

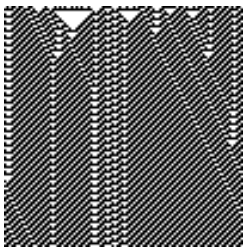

Rule 54


Rule 56

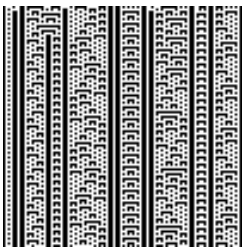

Rule 57


Rule 58

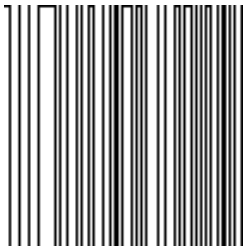

Rule 60


Rule 62

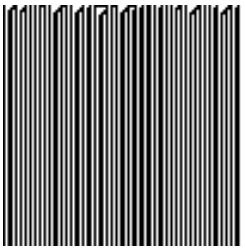

Rule 72


Rule 73

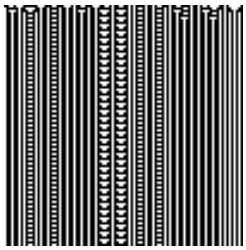

Rule 74


Rule 76

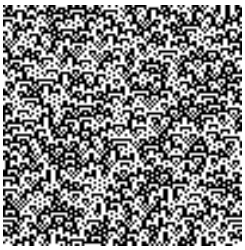

Rule 77

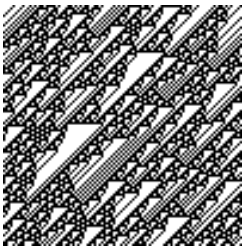

Rule 78

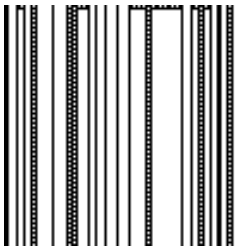

Rule 90

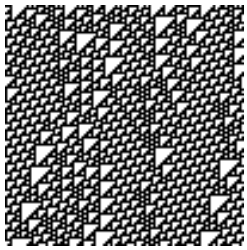

Rule 94

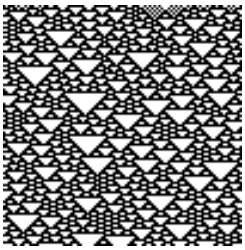

Rule 104

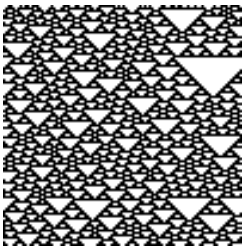

Rule 105


Rule 106

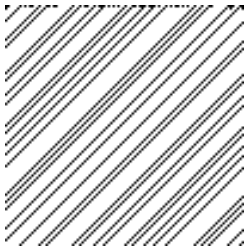

Rule 108

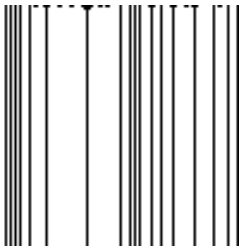

Rule 110

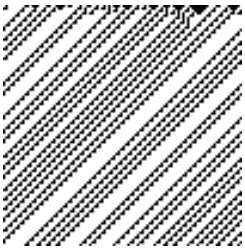

Rule 122


Rule 126

Rule 128

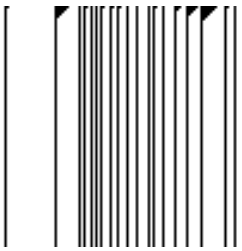

Rule 130

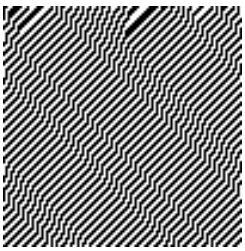

Rule 132


Rule 134

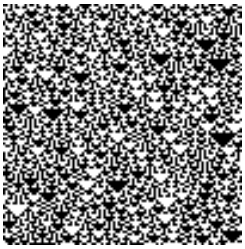

Rule 136

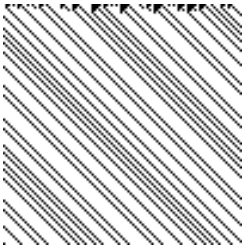

Rule 138

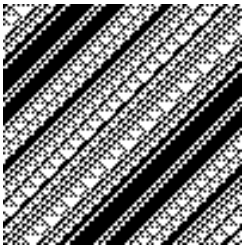

Rule 140

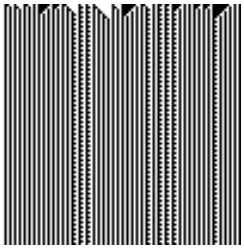

Rule 142


Rule 146


Rule 150

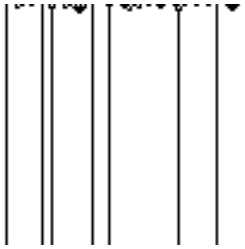

Rule 152


Rule 154


Rule 156


Rule 160

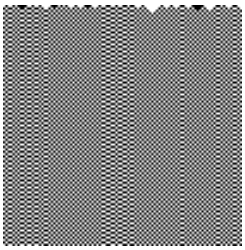

Rule 162

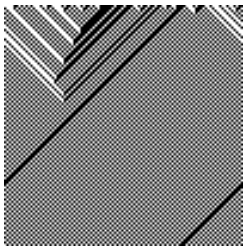

Rule 164

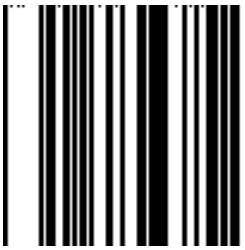

Rule 168

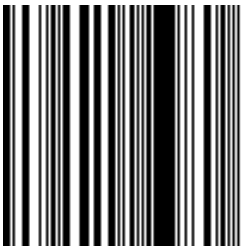

Rule 170


Rule 172


Rule 178


Rule 184


Rule 200


Rule 204


Rule 232

Single 1 histories

One method used to study these automata is to follow its history with an initial state of all 0s except for a single cell with a 1. When the rule number is even it makes sense to interpret state at each time, *t*, as an integer expressed in binary, producing a sequence *a*(*t*) of integers. In many cases these sequences have simple, closed form expressions or have a generating function with a simple form. The following rules are notable:

Rule 28

The sequence generated is 1, 3, 5, 11, 21, 43, 85, 171, ... (sequence OEIS A001045). This is the sequence of Jacobsthal numbers and has generating function

$$\frac{1+2x}{(1+x)(1-2x)}.$$

It has the closed form expression

$$a(t) = (4 \cdot 2^t - (-1)^t)/3$$

Note that rule 156 generates the same sequence.

Rule 50

The sequence generated is 1, 5, 21, 85, 341, 1365, 5461, 21845, ... (sequence OEIS A002450). This has generating function

$$\frac{1}{(1-x)(1-4x)}.$$

It has the closed form expression

$$a(t) = (4 \cdot 4^t - 1)/3.$$

Note that rules 58, 114, 122, 178, 186, 242 and 250 generate the same sequence.

Rule 54

The sequence generated is 1, 7, 17, 119, 273, 1911, 4369, 30583, ... (sequence OEIS A118108). This has generating function

$$\frac{1+7x}{(1-x^2)(1-16x^2)}.$$

It has the closed form expression

$$a(t) = (22 \cdot 4^t - 6(-4)^t - 4 + 3(-1)^t)/15.$$

Rule 60

The sequence generated is 1, 3, 5, 15, 17, 51, 85, 255, ... (sequence OEIS A001317). This can be obtained by taking successive rows of Pascal's triangle modulo 2 and interpreting them as integers in binary, which can be graphically represented by a Sierpinski triangle.

Rule 90

The sequence generated is 1, 5, 17, 85, 257, 1285, 4369, 21845, ... (sequence OEIS A038183). This can be obtained by taking successive rows of Pascal's triangle modulo 2 and interpreting them as integers in base 4. Note that rules 18, 26, 82, 146, 154, 210 and 218 generate the same sequence.

Rule 94

The sequence generated is 1, 7, 27, 119, 427, 1879, 6827, 30039, ... (sequence OEIS A118101). This can be expressed as

$$a(t) = \begin{cases} 1, & \text{if } t = 0 \\ 7, & \text{if } t = 1 \\ (1 + 5 \cdot 4^n)/3, & \text{if } t \text{ is even } > 0 \\ (10 + 11 \cdot 4^n)/6, & \text{if } t \text{ is odd } > 1 \end{cases}.$$

This has generating function

$$\frac{(1+2x)(1+5x-16x^4)}{(1-x^2)(1-16x^2)}.$$

Rule 102

The sequence generated is 1, 6, 20, 120, 272, 1632, 5440, 32640, ... (sequence OEIS A117998). This is simply the sequence generated by rule 60 (which is its mirror rule) multiplied by successive powers of 2.

Rule 150

The sequence generated is 1, 7, 21, 107, 273, 1911, 5189, 28123, ... (sequence OEIS A038184). This can be obtained by taking the coefficients of the successive powers of $(1+x+x^2)$ modulo 2 and interpreting them as integers in binary.

Rule 158

The sequence generated is 1, 7, 29, 115, 477, 1843, 7645, 29491, ... (sequence OEIS A118171). This has generating function

$$\frac{1+7x+12x^2-4x^3}{(1-x^2)(1-16x^2)}.$$

Rule 188

The sequence generated is 1, 3, 5, 15, 29, 55, 93, 247, ... (sequence OEIS A118173). This has generating function

$$\frac{1+3x+4x^2+12x^3+8x^4-8x^5}{(1-x^2)(1-16x^4)}.$$

Rule 190

The sequence generated is 1, 7, 29, 119, 477, 1911, 7645, 30583, ... (sequence OEIS A037576). This has generating function

$$\frac{1+3x}{(1-x^2)(1-4x)}.$$

Rule 220

The sequence generated is 1, 3, 7, 15, 31, 63, 127, 255, ... (sequence OEIS A000225). This is the sequence of Mersenne numbers and has generating function

$$\frac{1}{(1-x)(1-2x)}.$$

It has the closed form expression

$$a(t) = 2 \cdot 2^t - 1.$$

Note that rule 252 generates the same sequence.

Rule 222

The sequence generated is 1, 7, 31, 127, 511, 2047, 8191, 32767, ... (sequence OEIS A083420). This is every other entry in the sequence of Mersenne numbers and has generating function

$$\frac{1+2x}{(1-x)(1-4x)}.$$

It has the closed form expression

$$a(t) = 2 \cdot 4^t - 1.$$

Note that rule 254 generates the same sequence.

Random initial state

A second way to investigate the behavior of these automata is to examine its history starting with a random state. This behavior can be better understood in terms of Wolfram Classes. Wolfram gives the following examples as typical rules of each class.[1]

- Class 1: Cellular automata which rapidly converge to a uniform state. Examples are rules 0, 32, 160 and 250.
- Class 2: Cellular automata which rapidly converge to a repetitive or stable state. Examples are rules 4, 108, 218 and 232.
- Class 3: Cellular automata which appear to remain in a random state. Examples are rules 22, 30, 126, 150, 182.
- Class 4: Cellular automata which form areas of repetitive or stable states, but also form structures that interact with each other in complicated ways. An example is rule 110. Rule 110 has been shown to be capable of universal computation.[2]

Unusual cases

In some cases the behavior of a cellular automaton is not immediately obvious. For example, for Rule 62, interacting structures develop as in a Class 4. But in these interactions at least one of the structures is annihilated so the automaton eventually enters a repetitive state and the cellular automaton is Class 2.[3]

Rule 73 is Class 2[4] because any time there are two consecutive 1s surrounded by 0s, this feature is preserved in succeeding generations. This effectively creates walls which block the flow of information between different parts of the array. There are a finite number of possible configurations in the section between two walls so the automaton must eventually start repeating inside each section, though the period may be very long if the section is wide enough. These walls will form with probability 1 for completely random initial conditions. However, if the condition is added that the lengths of runs of consecutive 0s or 1s must always be odd, then the automaton displays Class 3 behavior since the walls can never form.

Rule 54 is Class 4[5] , but it remains unknown whether it is capable of universal computation. Interacting structures form, but structures that are useful for computation have yet to be found.[6]

References

- Weisstein, Eric W., "Elementary Cellular Automaton [7]" from MathWorld.
- Weisstein, Eric W., "Rule 30 [8]" from MathWorld.
- Weisstein, Eric W., "Rule 50 [9]" from MathWorld.
- Weisstein, Eric W., "Rule 54 [10]" from MathWorld.
- Weisstein, Eric W., "Rule 60 [11]" from MathWorld.
- Weisstein, Eric W., "Rule 62 [12]" from MathWorld.
- Weisstein, Eric W., "Rule 90 [13]" from MathWorld.
- Weisstein, Eric W., "Rule 94 [14]" from MathWorld.
- Weisstein, Eric W., "Rule 102 [15]" from MathWorld.
- Weisstein, Eric W., "Rule 110 [16]" from MathWorld.
- Weisstein, Eric W., "Rule 126 [17]" from MathWorld.
- Weisstein, Eric W., "Rule 150 [18]" from MathWorld.
- Weisstein, Eric W., "Rule 158 [19]" from MathWorld.
- Weisstein, Eric W., "Rule 182 [20]" from MathWorld.
- Weisstein, Eric W., "Rule 188 [21]" from MathWorld.
- Weisstein, Eric W., "Rule 190 [22]" from MathWorld.
- Weisstein, Eric W., "Rule 220 [23]" from MathWorld.
- Weisstein, Eric W., "Rule 222 [24]" from MathWorld.

[1] Stephan Wolfram, *A New Kind of Science* p223 ff.
[2] Rule 110 - Wolfram|Alpha (http://www30.wolframalpha.com/input/?i=rule+110)
[3] Rule 62 - Wolfram|Alpha (http://www30.wolframalpha.com/input/?i=rule+62)
[4] Rule 73 - Wolfram|Alpha (http://www30.wolframalpha.com/input/?i=rule+73)
[5] Rule 54 - Wolfram|Alpha (http://www30.wolframalpha.com/input/?i=rule+54)
[6] *A New Kind of Science* p697
[7] http://mathworld.wolfram.com/ElementaryCellularAutomaton.html
[8] http://mathworld.wolfram.com/Rule30.html
[9] http://mathworld.wolfram.com/Rule50.html
[10] http://mathworld.wolfram.com/Rule54.html
[11] http://mathworld.wolfram.com/Rule60.html
[12] http://mathworld.wolfram.com/Rule62.html
[13] http://mathworld.wolfram.com/Rule90.html
[14] http://mathworld.wolfram.com/Rule94.html
[15] http://mathworld.wolfram.com/Rule102.html
[16] http://mathworld.wolfram.com/Rule110.html
[17] http://mathworld.wolfram.com/Rule126.html
[18] http://mathworld.wolfram.com/Rule150.html
[19] http://mathworld.wolfram.com/Rule158.html
[20] http://mathworld.wolfram.com/Rule182.html
[21] http://mathworld.wolfram.com/Rule188.html
[22] http://mathworld.wolfram.com/Rule190.html
[23] http://mathworld.wolfram.com/Rule220.html
[24] http://mathworld.wolfram.com/Rule222.html

External links

"Elementary Cellular Automata" at the *Wolfram Atlas of Simple Programs* (http://atlas.wolfram.com/01/01)

Rule 30

Rule 30 is a one-dimensional binary cellular automaton rule introduced by Stephen Wolfram in 1983.[1] Wolfram describes it as being his "all-time favourite rule"[2] and details it in his book, *A New Kind of Science*. Using Wolfram's classification scheme, Rule 30 is a Class III rule, displaying aperiodic, chaotic behaviour.

A *Conus textile* shell similar in appearance to Rule 30.

This rule is of particular interest because it produces complex, seemingly random patterns from simple, well-defined rules. Because of this, Wolfram believes that rule 30, and cellular automata in general, are the key to understanding how simple rules produce complex structures and behaviour in nature. For instance, a pattern resembling Rule 30 appears on the shell of the widespread cone snail species *Conus textile*. Rule 30 has also been used as a random number generator in Wolfram's program Mathematica, and has also been proposed as a possible stream cipher for use in cryptography.[3] However, Sipper and Tomassini have shown that as a random number generator rule 30 exhibits poor behavior on a chi squared test compared to other cellular automaton based generators.[4]

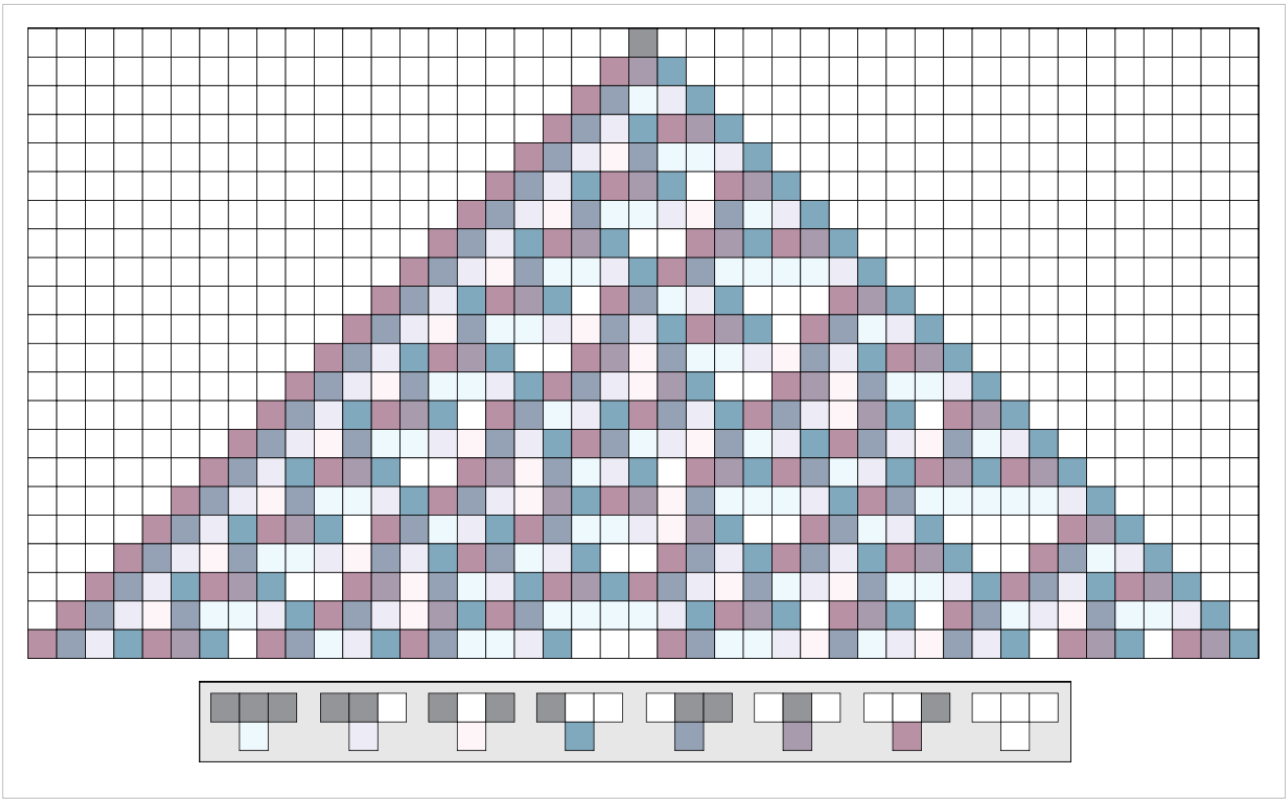
Rule 30 is so named because 30 is the smallest Wolfram code which describes its rule set (as described below). The mirror image, complement, and mirror complement of Rule 30 have Wolfram codes 86, 135, and 149, respectively.

Rule set

In all of Wolfram's elementary cellular automata, an infinite one-dimensional array of cellular automaton cells with only two states is considered, with each cell in some initial state. At discrete time intervals, every cell spontaneously changes state based on its current state and the state of its two neighbors. For Rule 30, the rule set which governs the next state of the automaton is:

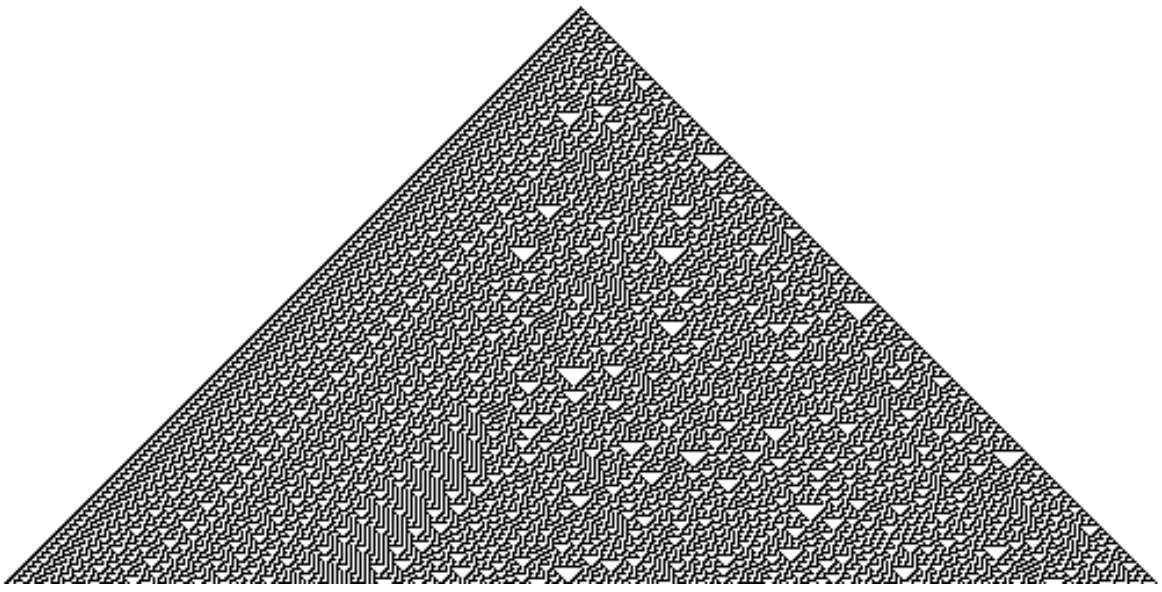
current pattern	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
new state for center cell	0	0	0	1	1	1	1	0

The following diagram shows the pattern created, with cells colored based on the previous state of their neighborhood. Darker colors represent "1" and lighter colors represent "0". Time increases down the vertical axis.



Structure and properties

The following pattern emerges from an initial state in a single cell with state 1 (shown as black) is surrounded by cells with state 0 (white).



Rule 30 cellular automaton

Here, the vertical axis represents time and any horizontal cross-section of the image represents the state of all the cells in the array at a specific point in the pattern's evolution. Several motifs are present in this structure, such as the frequent appearance of white triangles and a well-defined striped pattern on the left side; however the structure as a whole has no discernible pattern. The number of black cells at generation n is given by the sequence

1, 3, 3, 6, 4, 9, 5, 12, 7, 12, 11, 14, 12, 19, 13, 22, 15, 19, ... (sequence A070952)

and is approximately n .

As is apparent from the image above, rule 30 generates seeming randomness despite the lack of anything that could reasonably be considered random input. Stephen Wolfram proposed using its center column as a pseudorandom number generator (PRNG); it passes many standard tests for randomness, and Wolfram uses this rule in the Mathematica product for creating random integers. Although Rule 30 produces randomness on many input patterns, there are also an infinite number of input patterns that result in repeating patterns. The trivial example of such a pattern is the input pattern only consisting of zeros. A less trivial example, found by Matthew Cook, is any input pattern consisting of infinite repetitions of the pattern '00001000111000', with repetitions optionally being separated by six ones. Many more such patterns were found by Frans Faase. See Repeating Rule 30 patterns [5].

Chaos

Wolfram based his classification of Rule 30 as chaotic based primarily on its visual appearance, but it was later shown to meet more rigorous definitions of chaos proposed by Devaney and Knudson. In particular, according to Devaney's criteria, Rule 30 displays sensitive dependence on initial conditions (two initial configurations that differ only in a small number of cells rapidly diverge), its periodic configurations are dense in the space of all configurations, according to the Cantor topology on the space of configurations (there is a periodic configuration with any finite pattern of cells), and it is mixing (for any two finite patterns of cells, there is a configuration containing one pattern that eventually leads to a configuration containing the other pattern). According to Knudson's criteria, it displays sensitive dependence and there is a dense orbit (an initial configuration that eventually displays any finite pattern of cells). Both of these characterizations of the rule's chaotic behavior follow from a simpler and easy to verify property of Rule 30: it is *left permutative*, meaning that if two configurations `C` and `D` differ in the state of a single cell at position `i`, then after a single step the new configurations will differ at cell $i + 1$.[6]

External links

- Rule 30: Wolfram's Pseudo-random Bit Generator [7]. Recipe 32 at David Griffeath's Primordial Soup Kitchen.
- Weisstein, Eric W., "Rule 30 [8]" from MathWorld.
- Repeating Rule 30 patterns [5]. A list of patterns that, when repeated to fill the cells of a Rule 30 automaton, repeat themselves after finitely many time steps. Frans Faase, 2003.
- Paving Mosaic Fractal [8]. Basic introduction to the pattern of Rule 30 from the perspective of a LOGO software expert Olivier Schmidt-Chevalier.
- TED Talk from February 2010 [9]. Stephen Wolfram speaks about computing a theory of everything where he talks about rule 30 among other things.

References

[1] Wolfram, S. (1983). "Statistical mechanics of cellular automata". *Rev. Mod. Phys.* **55** (3): 601–644. Bibcode 1983RvMP...55..601W. doi:10.1103/RevModPhys.55.601.

[2] On Starting a Long-Term Company (http://www.stephenwolfram.com/publications/talks/ycombinatorschool/), S. Wolfram, 2005.

[3] Wolfram, S. (1985). "Cryptography with cellular automata" (http://www.springerlink.com/content/6lc1m67ec2ek0nk3/). *Proceedings of Advances in Cryptology - CRYPTO '85*. Lecture Notes in Computer Science 218, Springer-Verlag. pp. 429. . See also Meier, Willi; Staffelbach, Othmar (1991). "Analysis of pseudo random sequences generated by cellular automata" (http://www.springerlink.com/content/8cwan65384nphd0u/). *Advances in Cryptology: Proc. Workshop on the Theory and Application of Cryptographic Techniques, EUROCRYPT '91*. Lecture Notes in Computer Science 547, Springer-Verlag. pp. 186. .

[4] Sipper, Moshe; Tomassini, Marco (1996). "Generating parallel random number generators by cellular programming". *International Journal of Modern Physics C* **7** (2): 181–190. doi:10.1142/S012918319600017X.

[5] http://www.iwriteiam.nl/Rule30.html

[6] Cattaneo, Gianpiero; Finelli, Michele; Margara, Luciano (2000). "Investigating topological chaos by elementary cellular automata dynamics". *Theoretical Computer Science* **244** (1–2): 219–241. doi:10.1016/S0304-3975(98)00345-4. MR1774395.

[7] http://psoup.math.wisc.edu/archive/recipe32.html

[8] http://olivier.sc.free.fr/logosc/cubisme/pavages.html

[9] http://www.ted.com/talks/stephen_wolfram_computing_a_theory_of_everything.html

- Wolfram, Stephen, 1985, *Cryptography with Cellular Automata* (http://www.stephenwolfram.com/publications/articles/ca/85-cryptography/1/text.html), CRYPTO'85.

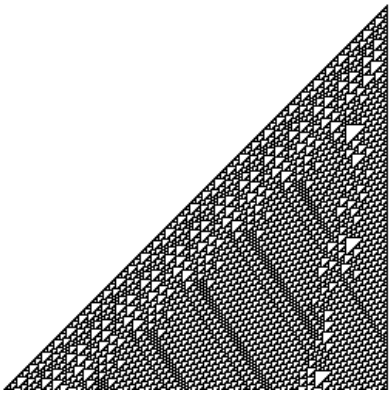
Rule 110

The **Rule 110 cellular automaton** (often simply **Rule 110**) is an elementary cellular automaton with the following rule table:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

In the table above, when the sequence of 1s and 0s corresponding to the new states is regarded as a binary number, the decimal equivalent is 110; hence the name of the rule.

History



An example run of a rule 110 cellular automaton

Around 2000, Matthew Cook published a proof of a 1985 conjecture by Stephen Wolfram by proving that Rule 110 is Turing complete, i.e., capable of universal computation. Cook presented his proof at the Santa Fe Institute conference CA98 before the publishing of Wolfram's book. This resulted in a legal affair based on a non-disclosure agreement with Wolfram Research. Wolfram Research blocked publication of Cook's proof for 2 years.[1]

Interesting properties

Among the 88 possible unique elementary cellular automata, Rule 110 is the only one for which this has been proven, although proofs for several similar rules should follow as simple corollaries, for instance Rule 124, where the only directional (asymmetrical) transformation is reversed. Rule 110 is arguably the simplest known Turing complete system.[1] [2]

Class 4 behavior

Rule 110, like the Game of Life, exhibits what Wolfram calls "Class 4 behavior", which is neither completely stable nor completely chaotic. Localized structures appear and interact in various complicated-looking ways.[3]

While working on the development of *NKS*, Wolfram's research assistant Matthew Cook proved Rule 110 capable of supporting universal computation. Rule 110 is a simple enough system to suggest that naturally occurring physical systems may also be capable of universality— meaning that many of their properties will be undecidable, and not amenable to closed-form mathematical solutions.[4]

Turing machine simulation overhead

The original emulation of a Turing machine contained an exponential time overhead due to the encoding of the Turing machine's tape using a unary numeral system. Neary and Woods (2006) modified the construction to use only a polynomial overhead.

The proof of universality

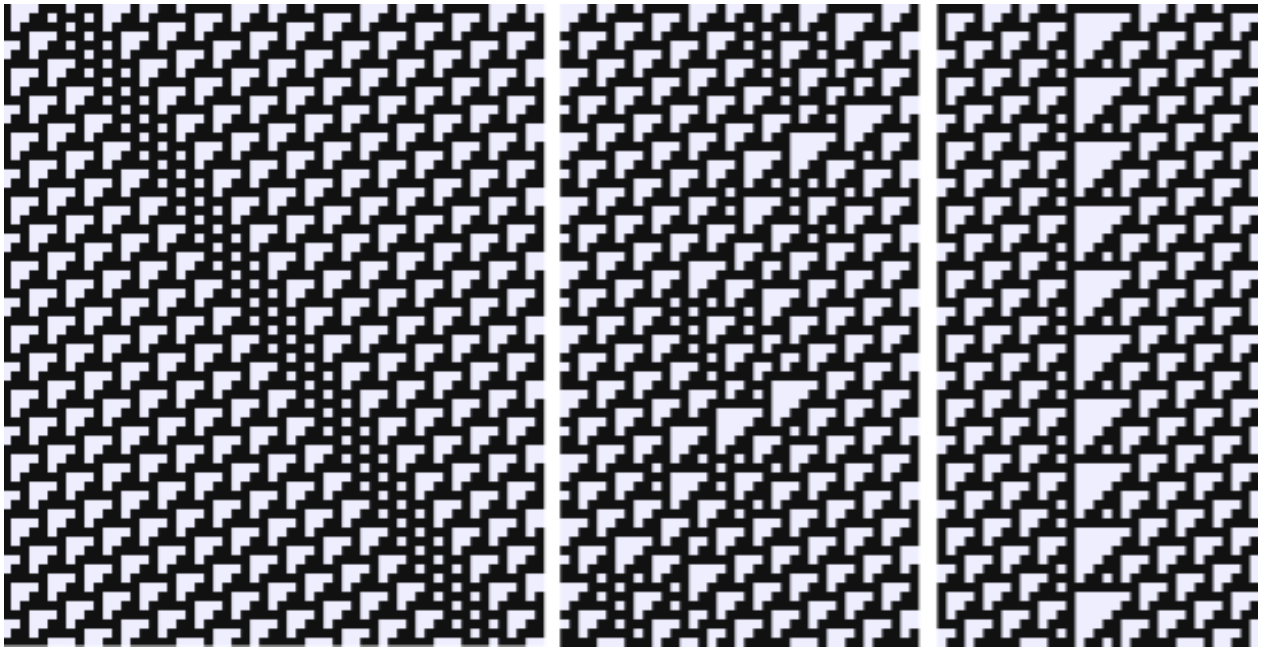
Matthew Cook presented his proof of the universality of Rule 110 at a Santa Fe Institute conference, held before the publication of *NKS*. Wolfram Research claimed that this presentation violated Cook's nondisclosure agreement with his employer, and obtained a court order excluding Cook's paper from the published conference proceedings. The existence of Cook's proof nevertheless became known. Interest in his proof stemmed not so much from its result as from its methods, specifically from the technical details of its construction . The character of Cook's proof differs considerably from the discussion of Rule 110 in *NKS*. Cook has since written a paper setting out his complete proof.[1]

Cook proved that Rule 110 was universal (or Turing complete) by showing it was possible to use the rule to emulate another computational model, the cyclic tag system, which is known to be universal. He first isolated a number of spaceships, self-perpetuating localized patterns, that could be constructed on an infinitely repeating pattern in a Rule 110 universe. He then devised a way for combinations of these structures to interact in a manner that could be exploited for computation.

Spaceships in Rule 110

The function of the universal machine in Rule 110 requires an infinite number of localized patterns to be embedded within an infinitely repeating background pattern. The background pattern is fourteen cells wide and repeats itself exactly every seven iterations. The pattern is **00010011011111**.

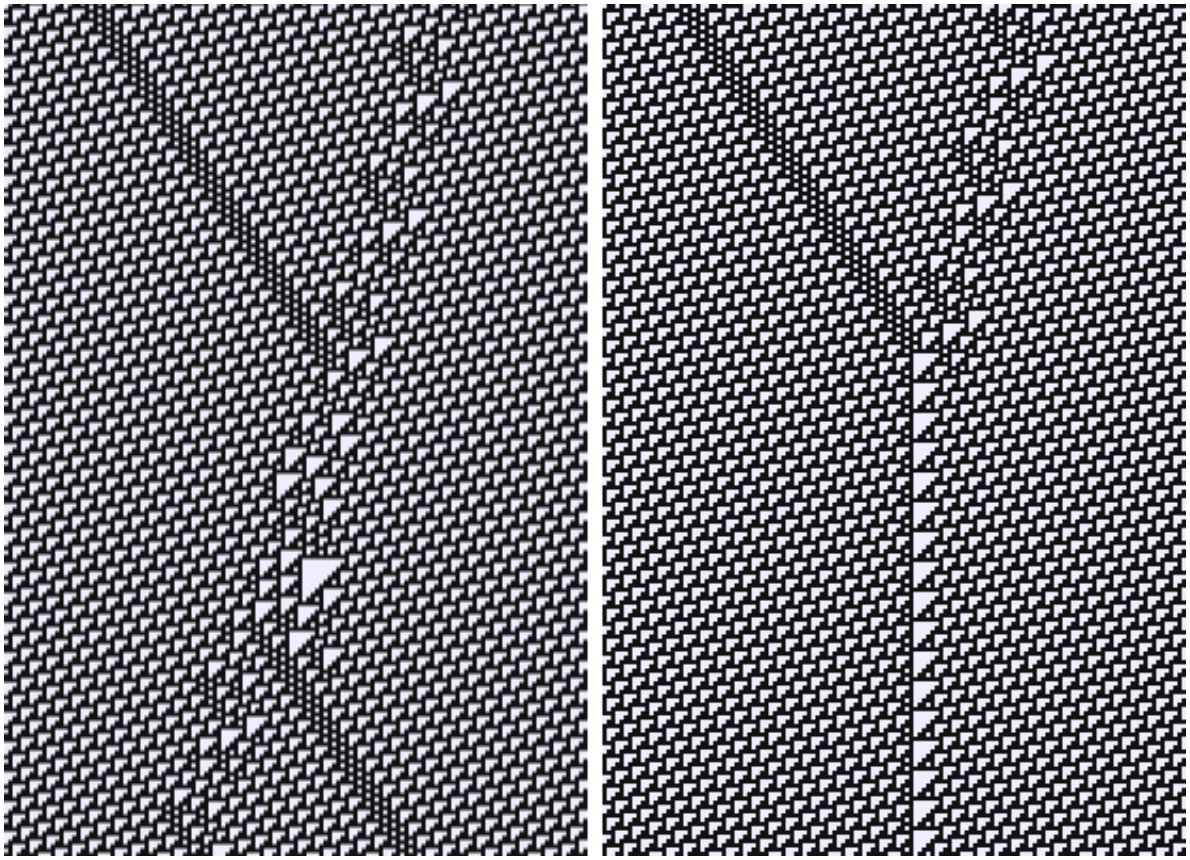
Three localized patterns are of particular importance in the Rule 110 universal machine. They are shown in the image below, surrounded by the repeating background pattern. The leftmost structure shifts to the right two cells and repeats every three generations. It comprises the sequence **0001110111** surrounded by the background pattern given above, as well as two different evolutions of this sequence.



The center structure shifts left eight cells and repeats every thirty generations. It comprises the sequence **1001111** surrounded by the background pattern given above, as well as twenty-nine different evolutions of this sequence.

The rightmost structure remains stationary and repeats every six generations. It comprises the sequence **111** surrounded by the background pattern given above, as well as five different evolutions of this sequence.

Below is an image showing the first two structures passing through each other without interacting (left), and interacting to form the third structure (right).



There are numerous other spaceships in Rule 110, but they do not feature as prominently in the universality proof.

Constructing the cyclic tag system

The cyclic tag system machinery has three main components:

- A *data string* which is stationary;
- An infinitely repeating series of finite *production rules* which start on the right and move leftward;
- An infinitely repeating series of *clock pulses* which start on the left and move rightward.

The initial spacing between these components is of utmost importance. In order for the cellular automaton to implement the cyclic tag system, the automaton's initial conditions must be carefully selected so that the various localized structures contained therein interact in a highly ordered way.

The *data string* in the cyclic tag system is represented by a series of stationary repeating structures of the type shown above. Varying amounts of horizontal space between these structures serve to differentiate **1** symbols from **0** symbols. These symbols represent the *word* on which the cyclic tag system is operating, and the first such symbol is destroyed upon consideration of every production rule. When this leading symbol is a **1**, new symbols are added to the end of the string; when it is **0**, no new symbols are added. The mechanism for achieving this is described below.

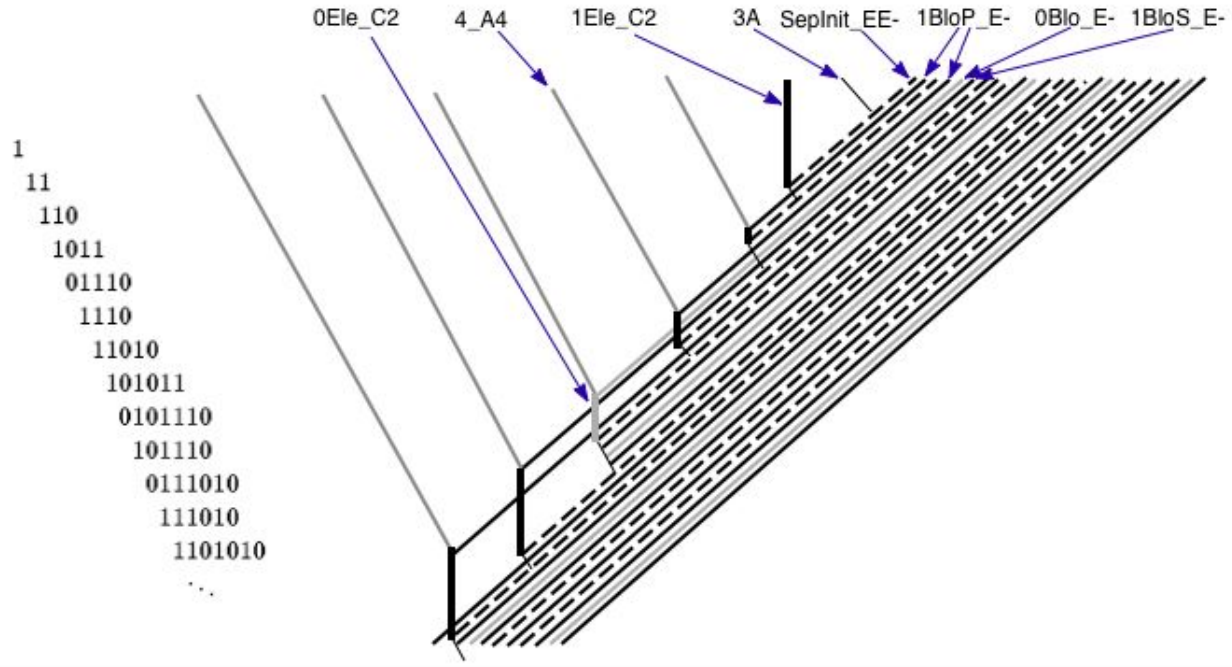
Entering from the right are a series of left-moving structures of the type shown above, separated by varying amounts of horizontal space. Large numbers of these structures are combined with different spacings to represent **0**s and **1**s in the cyclic tag system's production rules. Because the tag system's production rules are known at the time of creation of the program, and infinitely repeating, the patterns of **0**s and **1**s at the initial condition can be represented by an infinitely repeating string. Each production rule is separated from the next by another structure known as a *rule*

separator (or *block separator*), which moves towards the left at the same rate as the encoding of the production rules.

When a left-moving rule separator encounters a stationary symbol in the cyclic tag system's data string, it causes the first symbol it encounters to be destroyed. However, its subsequent behavior varies depending on whether the symbol encoded by the string had been a **0** or a **1**. If a **0**, the rule separator changes into a new structure which blocks the incoming production rule. This new structure is destroyed when it encounters the next rule separator.

If, on the other hand, the symbol in the string was a **1**, the rule separator changes into a new structure which admits the incoming production rule. Although the new structure is again destroyed when it encounters the next rule separator, it first allows a series of structures to pass through towards the left. These structures are then made to append themselves to the end of the cyclic tag system's data string. This final transformation is accomplished by means of a series of infinitely repeating, right-moving *clock pulses,* in the right-moving pattern shown above. The clock pulses transform incoming left-moving **1** symbols from a production rule into stationary **1** symbols of the data string, and incoming **0** symbols from a production rule into stationary **0** symbols of the data string.

Cyclic tag system working



The reconstructions was using a regular language to Rule 110 over an evolution space of 56,240 cells to 57,400 generations. Writing the sequence 1110111 on the tape of cyclic tag system and a leader component at the end with two solitons.

References

[1] *Complex Systems 15*, Issue 1, 2004. (http://www.complex-systems.com/Archive/hierarchy/abstract.cgi?vol=15&iss=1&art=01)
[2] Stephen Wolfram, *A New Kind of Science* p.169.
[3] Stephen Wolfram, *A New Kind of Science* p.229.
[4] Rule 110 – Wolfram Alpha (http://www.wolframalpha.com/input/?i=rule+110)

Bibliography

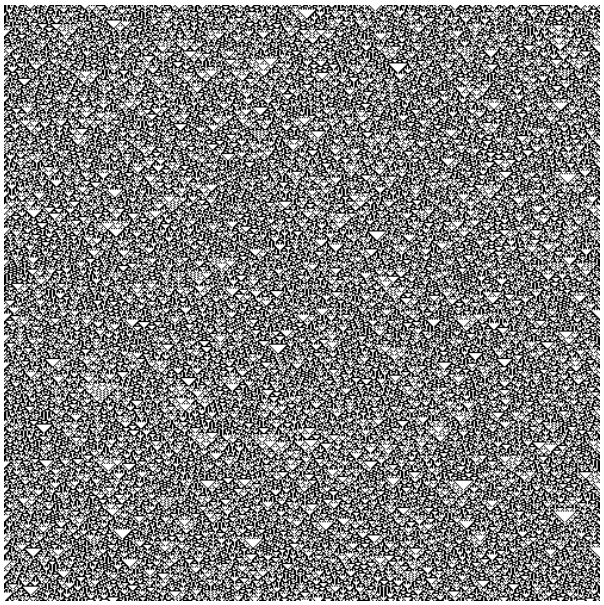
- Cook, Matthew (2004) " Universality in Elementary Cellular Automata (http://www.complex-systems.com/Archive/hierarchy/genlisting.cgi?vol=15&iss=1&vars=Menu_1_15=1=&Menu_1_14=0&label=Menu_1_15&state=0)", *Complex Systems 15*: 1-40.
- Martínez, Genaro J.; McIntosh, Harold V.; Mora, Seck Tuoh, Juan C. and Vergara, Sergio V. Chapa (2003-2008) " Reproducing the cyclic tag systems developed by Matthew Cook with Rule 110 using the phases fi_1 (http://uncomp.uwe.ac.uk/genaro/Papers/Papers_on_CA_files/repCTSR110.pdf)".
- Martínez, Genaro J.; McIntosh, Harold V.; Mora and Seck Tuoh, Juan C. and Vergara, Sergio V. Chapa (2008) " Determining a regular language by glider-based structures called phases fi_1 in Rule 110 (http://www.oldcitypublishing.com/JCA/JCA 3.3 contents.html)", *Journal of Cellular Automata 3 (3)*: 231-270.
- Martínez, Genaro J.; McIntosh, Harold V.; Mora, Seck Tuoh, Juan C. and Vergara, Sergio V. Chapa (2007) " Rule 110 objects and other constructions based-collisions (http://www.oldcitypublishing.com/JCA/JCA 2.3 contents.html)", *Journal of Cellular Automata 2 (3)*: 219-242.
- Martínez, Genaro J.; McIntosh, Harold V.; Mora and Seck Tuoh, Juan C. (2006) " Gliders in Rule 110 (http://www.oldcitypublishing.com/IJUC/IJUC 2.1 contents.html)", *Int. J. of Unconventional Computing 2*: 1-49.
- McIntosh, Harold V. (1999) " Rule 110 as it relates to the presence of gliders (http://delta.cs.cinvestav.mx/~mcintosh/comun/RULE110W/rule110.pdf)".
- McIntosh, Harold V. (2002) " Rule 110 Is Universal! (http://delta.cs.cinvestav.mx/~mcintosh/comun/texlet/texlet.pdf)".
- Neary, Turlough; and Woods, Damien (2006) " P-completeness of cellular automaton Rule 110 (http://citeseer.ist.psu.edu/neary06pcompleteness.html)", *Lecture Notes in Computer Science 4051*: 132-143.
- Stephen Wolfram (2002) *A New Kind of Science* (http://www.wolframscience.com/nksonline). Wolfram Media, Inc. ISBN 1-57955-008-8

External links

- For a nontechnical discussion of Rule 110 and its universality, see: A New Kind of Science p.690 (http://www.wolframscience.com/nksonline/page-690-text)
- Wolfram Science – Rule 110 (http://www.wolframalpha.com/input/?i=rule+110)

Rule 90

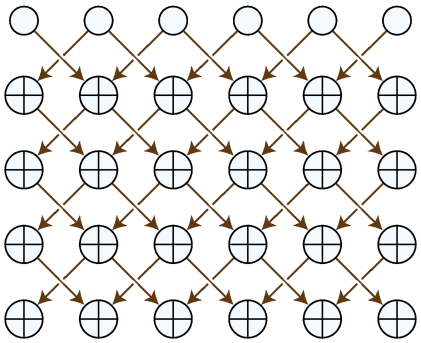
Rule 90 is an elementary cellular automaton based on the exclusive or function. It consists of a one-dimensional array of cells, each of which can hold either a 0 or a 1 value; in each time step all values are simultaneously replaced by the exclusive or of the two neighboring values.[1] Martin, Odlyzko & Wolfram (1984) call it "the simplest non-trivial cellular automaton",[2] and it is described extensively in Stephen Wolfram's 2002 book *A New Kind of Science*.[3] When started from a random initial configuration, its configuration remains random at each time step;[2] however, any configuration with only finitely many nonzero cells becomes a replicator that eventually fills all of the cells with copies of itself.[4]



Time-space diagram of Rule 90 with random initial conditions

Rules

An elementary cellular automaton consists of a one-dimensional array of cells, each of which holds a single binary value, either 0 or 1. An assignment of values to all of the cells is called a *configuration*. The automaton is given an initial configuration, after which its configuration repeatedly changes in a sequence of discrete time steps. At each step, all cells are updated simultaneously, according to a pre-specified rule which determines the new value as a function of each cell's previous value and of the values in its two neighboring cells. All cells obey the same rule, which may be given either as a formula or as a rule table that specifies the new value for each possible combination of neighboring values.[1]



In Rule 90, each cell's value is computed as the exclusive or of the two neighboring values in the previous time step.

In the case of Rule 90, each cell's new value is the exclusive or of the two neighboring values. Equivalently, the next state of this particular automaton is governed by the following rule table:[1]

current pattern	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
new state for center cell	0	1	0	1	1	0	1	0

Naming

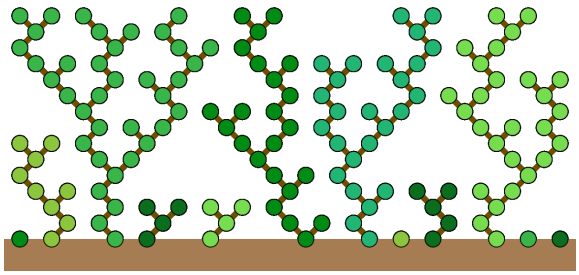
The name of Rule 90 comes from Stephen Wolfram's binary-decimal notation for one-dimensional cellular automaton rules. To calculate the notation for the rule, concatenate the new states in the rule table into a single binary number, and convert the number into decimal: $01011010_2 = 90_{10}$.[1] Rule 90 has also been called the **Sierpiński automaton**, due to the characteristic Sierpiński triangle shape it generates,[5] and the **Martin–Odlyzko–Wolfram cellular automaton** after the early research of Olivier Martin, Andrew M. Odlyzko, and Stephen Wolfram (1984) on this automaton.[6]

Additivity, superposition, and decomposition

A configuration in Rule 90 can be partitioned into two subsets of cells that do not interact with each other. One of these two subsets consists of the cells in even positions at even time steps and the cells in odd positions in odd time steps; the other subset consists of the cells in even positions at odd time steps and the cells in odd positions at even time steps. Each of these two subsets can be viewed as emulating a cellular automaton with only its half of the cells.[7]

Rule 90 is an *additive cellular automaton*: if two initial states are combined by computing the exclusive or of each their states, then their subsequent configurations will be combined in the same way. Thus, more generally, one can partition any configuration into two subsets with disjoint nonzero cells, evolve the two subsets separately, and compute the behavior of the original automaton by superposing configurations derived from the two subsets.[2]

Stunted trees and triangular clearings



A forest of stunted trees

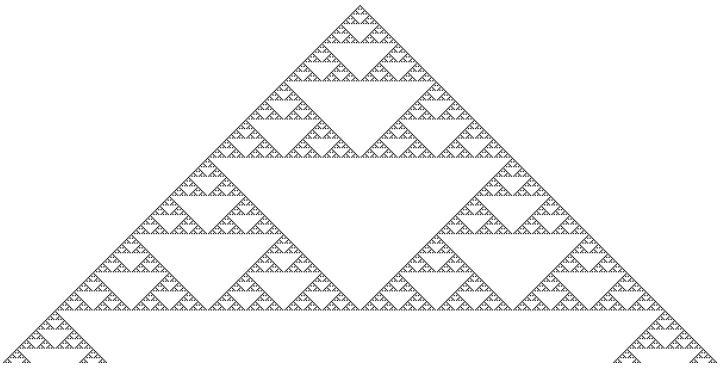
The Rule 90 automaton (in its equivalent form on one of the two independent subsets of alternating cells) was investigated in the early 1970s, in an attempt to gain additional insight into Gilbreath's conjecture on the differences of consecutive prime numbers: in the triangle of numbers generated from the primes by repeatedly applying the forward difference operator, most values are either 0 or 2, and Rule 90 describes the pattern of nonzeros that arises once all other values have been eliminated. Miller (1970) explained the rule by a metaphor of tree growth in a forest: each time step represents a height above the ground (with the initial configuration representing the ground level) and each nonzero cell represents a growing tree branch. At each successive level, a branch can grow into one of the cells only when there is no other branch competing for the same cell.[8]

From any initial configuration of Rule 90, one may form a mathematical forest, a directed acyclic graph in which every node has at most one outgoing edge, by creating a node for each pair (x,i) such that cell x is nonzero at time i, and by connecting each such node (with $i > 0$) to the unique nonzero neighbor of x in time step $i - 1$. Miller observed that these forests develop triangular "clearings", regions of the time-space diagram with no nonzero cells bounded by a flat bottom edge and diagonal sides. For random initial conditions, the boundaries between the trees formed in this way themselves shift in a seemingly random pattern, and trees frequently die out altogether. But by means of the theory of shift registers he and others were able to find initial conditions in which the trees all remain alive forever, the pattern of growth repeats periodically, and all of the clearings can be guaranteed to remain bounded in size.[8] [9]

Additionally, Miller used these regular patterns to form the designs of tapestries depicting physical trees as well as abstract patterns of triangles.[8]

Sierpiński triangle

The time-space diagram of Rule 90 (a plot in which the *i*th row records the configuration of the automaton at step *i*) has the appearance of the Sierpiński triangle fractal when responding to an initial state with a single nonzero cell. Rules 18, 22, 26, 82, 146, 154, 210 and 218 generate the same sequence. In Rule 90, each cell is the exclusive or of its two neighbors. Because this is equivalent to modulo-2 addition, this generates the modulo-2 version of Pascal's triangle, which is a discrete version of the Sierpiński triangle.[1] [10]



Sierpiński triangle generated by Rule 90

The number of live cells in the *i*th row of this pattern is 2^k, where k is the number of nonzero digits in the binary representation of the number i. The sequence of these numbers of live cells,

> 1, 2, 2, 4, 2, 4, 4, 8, 2, 4, 4, 8, 4, 8, 8, 16, 2, 4, 4, 8, 4, 8, 8, 16, 4, 8, 8, 16, 8, 16, 16, 32, ... (sequence OEIS A001316)

known as Gould's sequence or Dress's sequence, has a characteristic exponentially growing sawtooth shape that can be used to recognize physical processes that behave similarly to Rule 90.[5]

The Sierpiński triangle also occurs in a more subtle way in the evolution of any configuration in Rule 90. At any time step i in the Rule's evolution, each cell has a state that is the exclusive or of a subset of the cells in the initial configuration; that subset has the same shape as the *i*th row of the Sierpiński triangle.[11]

Replication

In the Sierpiński triangle, for any integer i, the rows with positions that are multiples of 2^i consist of some number of nonzero cells spaced 2^i units apart. Therefore, because of the additive property of Rule 90, if an initial configuration consists of a finite pattern P of nonzero cells with width less than 2^i, then in steps that are multiples of 2^i, the configuration will consist of copies of P spaced 2^i units from start to start; this spacing is wide enough to prevent the copies from interfering with each other. The number of copies is the same as the number of nonzero cells in the corresponding row of the Sierpiński triangle. Thus, in this rule, every pattern is a replicator: it generates multiple copies of itself that spread out across the configuration, eventually filling the whole array. However, unlike the replicators in more complex rules such as the Von Neumann universal constructor, Codd's cellular automaton, or Langton's loops, the replication in Rule 90 is trivial and automatic, rather than requiring each replicator pattern to carry and copy a sequence of instructions for building itself.[4]

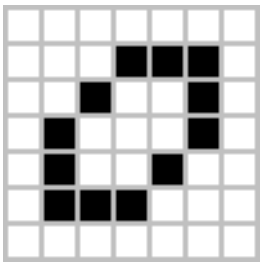
Predecessors and Gardens of Eden

In Rule 90, on an infinite one-dimensional lattice, every configuration has exactly four predecessor configurations: in the predecessor, any two consecutive cells may have any combination of states, but once those two cells' states are chosen, there is only one consistent choice for the states of the remaining cells. Therefore, this rule provides an example of a cellular automaton that is surjective (each configuration has a predecessor, so there is no Garden of Eden) but not injective (unlike injective rules, Rule 90 has pairs of configurations with the same successor). The Garden of Eden theorem of Moore and Myhill implies that every injective cellular automaton must be surjective, but this example shows that the converse is not true.[12] [13]

Because each state has a bounded number of predecessors, the evolution of Rule 90 preserves the entropy of any configuration. In particular, if an infinite initial configuration is selected by choosing the state of each cell independently at random, with each of the two states being equally likely to be selected, then each subsequent configuration can be described by exactly the same probability distribution.[2]

The Rule 90 configuration consisting of a single nonzero cell (with all other cells zero) has no predecessor that have finitely many nonzeros, but it is not a Garden of Eden because it has predecessors with infinitely many nonzeros.[12]

Emulation by other systems



The bowtie pasta replicator in HighLife, one-dimensional arrays of which can be used to emulate Rule 90

Many other cellular automata and other computational systems are capable of emulating the behavior of Rule 90. For instance, a configuration in rule 90 may be translated into a configuration into the different elementary cellular automaton Rule 22 by replacing each Rule 90 cell by three consecutive Rule 22 cells that are either all zero (if the Rule 90 cell is itself zero) or a one followed by two zeros (if the Rule 90 cell is a one). With this transformation, every six steps of the Rule 22 automaton simulates a single step of the Rule 90 automaton. Similar direct simulations of Rule 90 are also possible for the elementary cellular automata Rule 45 and Rule 126, for certain string rewriting systems and tag systems, and in two-dimensional cellular automata including Wireworld. Rule 90 can also simulate itself in the same way: if each cell of a Rule 90 configuration is replaced by a pair of consecutive cells, in which the first cell in the pair contains the original cell's value, and the second contains a zero, then this doubled configuration has the same behavior as the original configuration at half the speed.[14]

Various other cellular automata are known to support replicators, patterns that make copies of themselves, with the same behavior as in the tree growth model for Rule 90: a new copy is placed to either side of the replicator pattern, as long as the space there is empty, but if two replicators both attempt to copy themselves into the same position, then the space remains blank, and in either case the replicators themselves vanish, leaving their copies to carry on the replication. A standard example of this behavior is the "bowtie pasta" pattern in the two-dimensional HighLife rule, a rule that except for the behavior of this pattern behaves in many ways like Conway's Game of Life. Whenever a pattern supports replicators with this growth pattern, one-dimensional arrays of replicators can be used to simulate Rule 90.[15] Rule 90 (on finite rows of cells) can also be simulated by the block oscillators of the two-dimensional Life-like cellular automaton B36/S125, also called "2x2", and the behavior of Rule 90 can be used to characterize the possible periods of these oscillators.[16]

References

[1] Wolfram, Stephen (1983), "Statistical mechanics of cellular automata" (http://www.stephenwolfram.com/publications/articles/ca/83-statistical/), *Reviews of Modern Physics* **55** (3): 601–644, Bibcode 1983RvMP...55..601W, doi:10.1103/RevModPhys.55.601, .

[2] Martin, Olivier; Odlyzko, Andrew M.; Wolfram, Stephen (1984), "Algebraic properties of cellular automata" (http://www.stephenwolfram.com/publications/articles/ca/84-properties/), *Communications in Mathematical Physics* **93** (2): 219–258, doi:10.1007/BF01223745, .

[3] Wolfram, Stephen (2002), *A New Kind of Science*, Wolfram Media. The book's index lists over 50 distinct subtopics for Rule 90.

[4] Waksman, Abraham (1969), "A model of replication", *Journal of the ACM* **16** (1): 178–188, doi:10.1145/321495.321509; Amoroso, Serafino; Cooper, Gerald (1971), "Tessellation structures for reproduction of arbitrary patterns", *Journal of Computer and System Sciences* **5** (5): 455–464, doi:10.1016/S0022-0000(71)80009-0. Wolfram (1983) (Fig.33 and surrounding text) also mentions the same property, and as well as citing Waksman, Amoroso, and Cooper he credits its observation to unpublished work by Edward Fredkin in 1981.

[5] Claussen, Jens Christian; Nagler, Jan; Schuster, Heinz Georg (2004), "Sierpinski signal generates $1/f^{\alpha}$ spectra", *Physical Review E* **70**: 032101, arXiv:cond-mat/0308277, doi:10.1103/PhysRevE.70.032101.

[6] Misiurewicz, Michał; Stevens, John G.; Thomas, Diana M. (2006), "Iterations of linear maps over finite fields", *Linear Algebra and its Applications* **413** (1): 218–234, doi:10.1016/j.laa.2005.09.002.

[7] McIntosh, Harold V. (1993), *Ancestors: Commentaries on "The Global Dynamics of Cellular Automata" by Andrew Wuensche and Mike Lesser (Addison-Wesley, 1992)* (http://delta.cs.cinvestav.mx/~mcintosh/comun/wandl/global.pdf), Instituto de Ciencias, Universidad

Autónoma de Puebla, .

[8] Miller, J. C. P. (1970), "Periodic forests of stunted trees", *Philosophical Transactions of the Royal Society of London*, Series A, Mathematical and Physical Sciences **266** (1172): 63–111, JSTOR 73779.

[9] ApSimon, H. G. (1970), "Periodic forests whose largest clearings are of size 3", *Philosophical Transactions of the Royal Society of London*, Series A, Mathematical and Physical Sciences **266** (1172): 113–121, JSTOR 73780; ApSimon, H. G. (1970), "Periodic forests whose largest clearings are of size $n \geq 4$", *Philosophical Transactions of the Royal Society of London*, Series A, Mathematical and Physical Sciences **266** (1538): 399–404, JSTOR 73780. A similar analysis of periodic configurations in Rule 90 also appears in Wolfram (2002), p. 954.

[10] Wolfram (2002), pp. 25–26, 270–271, 870.

[11] Kar, B. K.; Gupta, A.; Chaudhuri, P. Pal (1993), "On explicit expressions in additive cellular automata theory", *Information Sciences* **72** (1–2): 83–103, doi:10.1016/0020-0255(93)90030-P.

[12] Skyum, Sven (1975), "Confusion in the Garden of Eden", *Proceedings of the American Mathematical Society* **50** (1): 332–336, doi:10.1090/S0002-9939-1975-0386350-1

[13] Sutner, Klaus (1991), "De Bruijn Graphs and Linear Cellular Automata" (http://www.complex-systems.com/pdf/05-1-3.pdf), *Complex Systems* **5**: 19–30, . Wolfram (2002), pp. 959–960. Martin, Odlyzko & Wolfram (1984) provide a similar analysis of the predecessors of the same rule for finite sets of cells with periodic boundary conditions.

[14] Wolfram (2002), pp. 269–270, 666–667, 701–702, 1117.

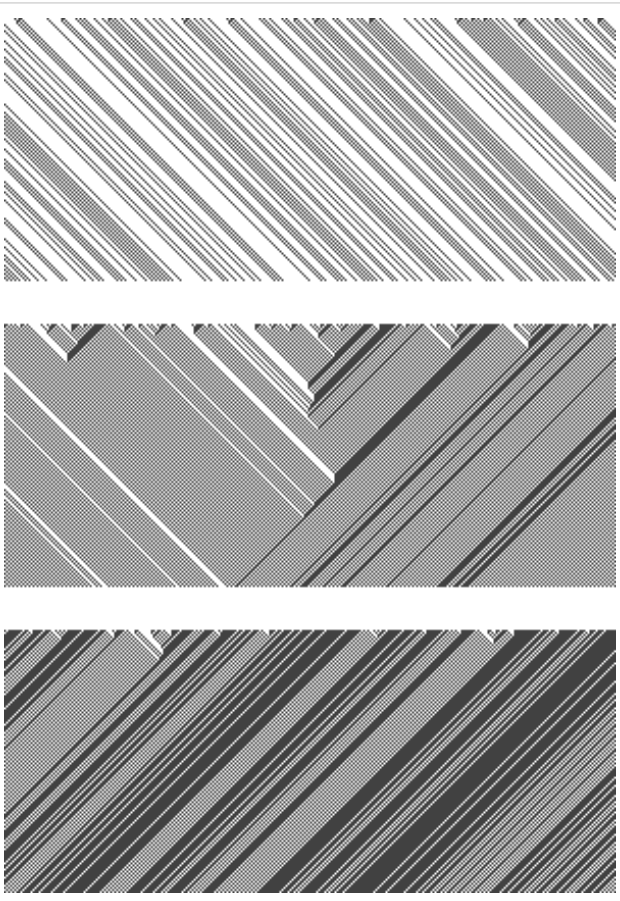
[15] Griffeath, David (1996), "Recipe for the week of July 1–7: Replicating Skeeters" (http://psoup.math.wisc.edu/archive/recipe75.html), *The Primordial Soup Kitchen*, .

[16] Johnston, Nathaniel (2010), "The B36/S125 "2x2" Life-like cellular automaton", in Adamatzky, Andrew, *Game of Life Cellular Automata*, Springer-Verlag, pp. 99–114.

Rule 184

Rule 184 is a one-dimensional binary cellular automaton rule, notable for solving the majority problem as well as for its ability to simultaneously describe several, seemingly quite different, particle systems:

- Rule 184 can be used as a simple model for traffic flow in a single lane of a highway, and forms the basis for many cellular automaton models of traffic flow with greater sophistication. In this model, particles (representing vehicles) move in a single direction, stopping and starting depending on the cars in front of them. The number of particles remains unchanged throughout the simulation. Because of this application, Rule 184 is sometimes called the "traffic rule".[1]
- Rule 184 also models a form of deposition of particles onto an irregular surface, in which each local minimum of the surface is filled with a particle in each step. At each step of the simulation, the number of particles increases. Once placed, a particle never moves.
- Rule 184 can be understood in terms of ballistic annihilation, a system of particles moving both leftwards and rightwards through a one-dimensional medium. When two such particles collide, they annihilate each other, so that at each step the number of particles remains unchanged or decreases.



Rule 184, run for 128 steps from random configurations with each of three different starting densities: top 25%, middle 50%, bottom 75%. The view shown is a 300-pixel crop from a wider simulation.

The apparent contradiction between these descriptions is resolved by different ways of corresponding features of the automaton's state with particles. The name of the rule is a Wolfram code that defines the evolution of its states. The earliest research on Rule 184 seems to be the papers by Li (1987) and Krug and Spohn (1988). In particular, Krug and Spohn already describe all three types of particle system modeled by Rule 184.[2]

Definition

A state of the Rule 184 automaton consists of a one-dimensional array of cells, each containing a binary value (0 or 1). In each step of its evolution, the Rule 184 automaton applies the following rule to each of the cells in the array, simultaneously for all cells, to determine the new state of the cell:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	1	0	1	1	1	0	0	0

An entry in this table defines the new state of each cell as a function of the previous state and the previous values of the neighboring cells on either side.

The name for this rule, Rule 184, is the Wolfram code describing the state table above: the bottom row of the table, 10111000, when viewed as a binary number, is equal to the decimal number 184.

The rule set for Rule 184 may also be described intuitively, in several different ways:

- At each step, whenever there exists in the current state a 1 immediately followed by a 0, these two symbols swap places. Based on this description, Krug and Spohn (1984) call Rule 184 a deterministic version of a "kinetic Ising model with asymmetric spin-exchange dynamics".
- At each step, if a cell with value 1 has a cell with value 0 immediately to its right, the 1 moves rightwards leaving a 0 behind. A 1 with another 1 to its right remains in place, while a 0 that does not have a 1 to its left stays a 0. This description is most apt for the application to traffic flow modeling.
- If a cell has state 0, its new state is taken from the cell to its left. Otherwise, its new state is taken from the cell to its right. That is, each cell can be implemented by a multiplexer, and is closely related in its operation to a Fredkin gate.[3]

Dynamics and majority classification

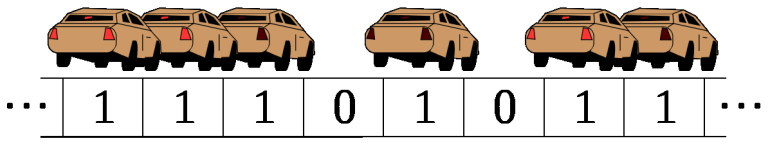
From the descriptions of the rules above, two important properties of its dynamics may immediately be seen. First, in Rule 184, for any finite set of cells with periodic boundary conditions, the number of 1s and the number of 0s in a pattern remains invariant throughout the pattern's evolution. Similarly, if the density of 1s is well-defined for an infinite array of cells, it remains invariant as the automaton carries out its steps.[4] And second, although Rule 184 is not symmetric under left-right reversal, it does have a different symmetry: reversing left and right and at the same time swapping the roles of the 0 and 1 symbols produces a cellular automaton with the same update rule.

Patterns in Rule 184 typically quickly stabilize, either to a pattern in which the cell states move in lockstep one position leftwards at each step, or to a pattern that moves one position rightwards at each step.[5] Specifically, if the initial density of cells with state 1 is less than 50%, the pattern stabilizes into clusters of cells in state 1, spaced two units apart, with the clusters separated by blocks of cells in state 0. Patterns of this type move rightwards. If, on the other hand, the initial density is greater than 50%, the pattern stabilizes into clusters of cells in state 0, spaced two units apart, with the clusters separated by blocks of cells in state 1, and patterns of this type move leftwards. If the density is exactly 50%, the initial pattern stabilizes (more slowly) to a pattern that can equivalently be viewed as moving either leftwards or rightwards at each step: an alternating sequence of 0s and 1s.

One can view Rule 184 as solving the majority problem, of constructing a cellular automaton that can determine whether an initial configuration has a majority of its cells active: if Rule 184 is run on a finite set of cells with periodic boundary conditions, and the number of active cells is less than half of all cells, then each cell will

eventually see two consecutive zero states infinitely often, and two consecutive one states only finitely often, while if the number of active cells forms a majority of the cells then each cell will eventually see two consecutive ones infinitely often and two consecutive zeros only finitely often.[6] The majority problem cannot be solved perfectly if it is required that all cells eventually stabilize to the majority state[7] but the Rule 184 solution avoids this impossibility result by relaxing the criterion by which the automaton recognizes a majority.

Traffic flow



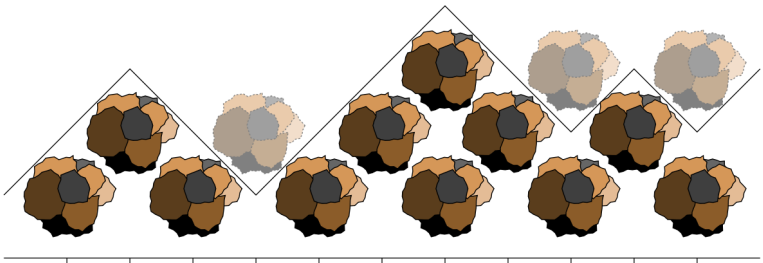
Rule 184 interpreted as a simulation of traffic flow. Each 1 cell corresponds to a vehicle, and each vehicle moves forwards only if it has open space in front of it.

If we interpret each 1 cell in Rule 184 as containing a particle, these particles behave in many ways similarly to automobiles in a single lane of traffic: they move forwards at a constant speed if there is open space in front of them, and otherwise they stop. Traffic models such as Rule 184 and its generalizations that discretize both space and time are commonly called *particle-hopping models*.[8] Although very primitive, the Rule 184 model of traffic flow already predicts some of the familiar emergent features of real traffic: clusters of freely moving cars separated by stretches of open road when traffic is light, and waves of stop-and-go traffic when it is heavy.[9]

It is difficult to pinpoint the first use of Rule 184 for traffic flow simulation, in part because the focus of research in this area has been less on achieving the greatest level of mathematical abstraction and more on verisimilitude: even the earlier papers on cellular automaton based traffic flow simulation typically make the model more complex in order to more accurately simulate real traffic. Nevertheless, Rule 184 is fundamental to traffic simulation by cellular automata. Wang et al. (1997), for instance, state that "the basic cellular automaton model describing a one-dimensional traffic flow problem is rule 184." Nagel (1996) writes "Much work using CA models for traffic is based on this model." Several authors describe one-dimensional models with vehicles moving at multiple speeds; such models degenerate to Rule 184 in the single-speed case.[10] Gaylord and Nishidate (1996) extend the Rule 184 dynamics to two-lane highway traffic with lane changes; their model shares with Rule 184 the property that it is symmetric under simultaneous left-right and 0-1 reversal. Biham et al. (1992) describe a two-dimensional city grid model in which the dynamics of individual lanes of traffic is essentially that of Rule 184.[11] For an in-depth survey of cellular automaton traffic modeling and associated statistical mechanics, see Chowdhury et al. (2000).

When viewing Rule 184 as a traffic model, it is natural to consider the average speed of the vehicles. When the density of traffic is less than 50%, this average speed is simply one unit of distance per unit of time: after the system stabilizes, no car ever slows. However, when the density is a number ρ greater than 1/2, the average speed of traffic is $\frac{1-\rho}{\rho}$. Thus, the system exhibits a second-order kinetic phase transition at ρ = 1/2. At that critical value the average speed approaches its stationary limit as the square root of the number of steps, while away from the critical value the approach is exponential.[12]

Surface deposition

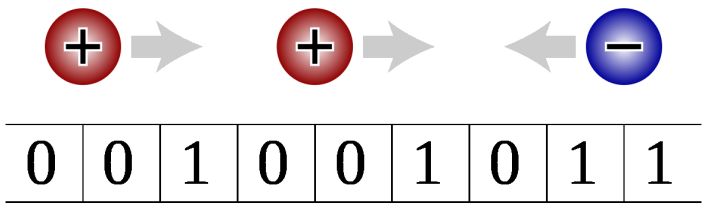


Rule 184 as a model of surface deposition. In a layer of particles forming a diagonally-oriented square lattice, new particles stick in each time step to the local minima of the surface. The cellular automaton states model the local slope of the surface.

As shown in the figure, and as originally described by Krug and Spohn (1988),[13] Rule 184 may be used to model deposition of particles onto a surface. In this model, one has a set of particles forming a surface, where the particles assume positions in a square lattice oriented diagonally. If a particle is present at some position of the lattice, the lattice positions below and to the right, and below and to the left of the particle must also be filled. At each time step, the surface grows by the deposition of particles in each local minimum of the surface; that is, at each position where it is possible to add a new particle. To model this process by Rule 184, we observe that the boundary between filled and unfilled lattice positions can be marked by a polygonal line, the segments of which separate adjacent lattice positions and have slopes +1 and -1. We model a segment with slope +1 by an automaton cell with state 0, and a segment with slope -1 by an automaton cell with state 1. The local minima of the surface are the points where a segment of slope -1 lies to the left of a segment of slope +1; that is, in the automaton, a position where a cell with state 1 lies to the left of a cell with state 0. Adding a particle to that position corresponds to changing the states of these two adjacent cells from 1,0 to 0,1, which is exactly the behavior of Rule 184.

Related work on this model concerns deposition in which the arrival times of additional particles are random, rather than having particles arrive at all local minima simultaneously.[14] These stochastic growth processes can be modeled as an asynchronous cellular automaton.

Ballistic annihilation



Rule 184 as a model of ballistic annihilation. Particles and antiparticles (modeled by consecutive cells with the same state) move in opposite directions and annihilate each other when they collide.

Ballistic annihilation describes a process by which moving particles and antiparticles annihilate each other when they collide. In the simplest version of this process, the system consists of a single type of particle and antiparticle, moving at equal speeds in opposite directions in a one-dimensional medium.

This process can be modeled by Rule 184, as follows. We view the particles as aligned, not with the cells of the automaton, but rather with the interstices between cells. If two consecutive cells both have state 0, we view the transition between them as forming a particle that moves rightwards one cell at each time step. If, on the other hand, two consecutive cells both have state 1, we view the transition between them as forming an antiparticle that moves leftwards one cell at each time step. Two consecutive cells with differing states are considered to form part of a background material without any particles in it. It may be observed that these particles interact by ballistic annihilation: when a rightwards-moving particle and a leftwards-moving antiparticle meet, the result is a region of background from which both particles have vanished, without any effect on any other nearby particles.[15]

The behavior of certain other systems, such as one-dimensional cyclic cellular automata, can also be described in terms of ballistic annihilation.[16] There is a technical restriction on the particle positions for the ballistic annihilation view of Rule 184 that does not arise in these other systems, stemming from the alternating pattern of the background: in the particle system corresponding to a Rule 184 state, if two consecutive particles are both of the same type they must be an odd number of cells apart, while if they are of opposite types they must be an even number of cells apart. However this parity restriction does not play a role in the statistical behavior of this system.

Pivato (2005) uses a similar but more complicated particle-system view of Rule 184: he not only views alternating 0-1 regions as background, but also considers regions consisting solely of a single state to be background as well. Based on this view he describes seven different particles formed by boundaries between regions, and classifies their possible interactions. See Chopard and Droz (1998, pp. 188–190) for a more general survey of the physics of annihilation processes.

Context free parsing

In his book *A New Kind of Science*, Stephen Wolfram points out that rule 184, when run on patterns with density 50%, can be interpreted as parsing the context free language describing strings formed from nested parentheses. This interpretation is closely related to the ballistic annihilation view of rule 184: in Wolfram's interpretation, an open parenthesis corresponds to a left-moving particle while a close parenthesis corresponds to a right-moving particle.

Notes

[1] E.g. see Fukś (1997).

[2] One can find many later papers that, when mentioning Rule 184, cite the early papers of Stephen Wolfram. However, Wolfram's papers consider only automata that are symmetric under left-right reversal, and therefore do not describe Rule 184.

[3] Li (1992). Li used this interpretation as part of a generalization of Rule 184 to nonlocal neighborhood structures.

[4] Boccara and Fukś (1998) have investigated more general automata with similar conservation properties, as has Moreira (2003).

[5] Li (1987).

[6] Capcarrere et al. (1996); Fukś (1997); Sukumar (1998).

[7] Land and Belew (1995).

[8] Nagel (1996); Chowdhury et al. (2000).

[9] Tadaki and Kikuchi (1994).

[10] For several models of this type see Nagel and Schreckenberg (1992), Fukui and Ishibashi (1996), and Fukś and Boccara (1998). Nagel (1996) observes the equivalence of these models to rule 184 in the single-speed case and lists several additional papers on this type of model.

[11] See also Tadaki and Kikuchi (1994) for additional analysis of this model.

[12] Fukś and Boccara (1998).

[13] See also Belitzky and Ferrari (1995) and Chopard and Droz (1998, p. 29).

[14] Also discussed by Krug and Spohn (1988).

[15] Krug and Spohn (1988); Belitzky and Ferrari (1995).

[16] Belitzky and Ferrari (1995).

References

- Belitsky, Vladimir; Ferrari, Pablo A. (1995). "Ballistic annihilation and deterministic surface growth". *Journal of Statistical Physics* **80** (3–4): 517–543. doi:10.1007/BF02178546.
- Biham, Ofer; Middleton, A. Alan; Levine, Dov (1992). "Self-organization and a dynamic transition in traffic-flow models". *Physical Review A* **46** (10): R6124–R6127. doi:10.1103/PhysRevA.46.R6124. PMID 9907993.
- Boccara, Nino; Fukś, Henryk (1998). "Cellular automaton rules conserving the number of active sites". *J. Phys. A: Math. Gen.* **31** (28): 6007–6018. arXiv:adap-org/9712003. doi:10.1088/0305-4470/31/28/014.
- Capcarrere, Mathieu S.; Sipper, Moshe; Tomassini, Marco (1996). "Two-state, $r = 1$ cellular automaton that classifies density" (http://www.leitl.org/docs/density.ps.gz). *Phys. Rev. Lett.* **77** (24): 4969. Bibcode 1996PhRvL..77.4969C. doi:10.1103/PhysRevLett.77.4969. PMID 10062680.

- Chopard, Bastien; Droz, Michel (1998). *Cellular Automata Modeling of Physical Systems*. Cambridge University Press. ISBN 0-521-67345-3.
- Chowdhury, Debashish; Santen, Ludger; Schadschneider, Andreas (2000). "Statistical physics of vehicular traffic and some related systems". *Physics Reports* **329** (4): 199–329. arXiv:cond-mat/0007053. doi:10.1016/S0370-1573(99)00117-9.
- Fukś, Henryk (1997). "Solution of the density classification problem with two similar cellular automata rules". *Physical Review E* **55** (3): R2081–R2084. doi:10.1103/PhysRevE.55.R2081.
- Fukś, Henryk; Boccara, Nino (1998). "Generalized deterministic traffic rules" (http://www.tjhsst.edu/~ekoniecz/fuks.pdf). *Journal of Modern Physics C* **9** (1): 1–12. doi:10.1142/S0129183198000029.
- Fukui, M.; Ishibashi, Y. (1996). "Traffic flow in 1D cellular automaton model including cars moving with high speed". *J. Phys. Soc. Japan* **65** (6): 1868–1870. doi:10.1143/JPSJ.65.1868.
- Gaylord, Richard J.; Nishidate, Kazume (1996). *Modeling Nature: Cellular Automata Simulations with Mathematica*. Springer-Verlag. pp. 29–35. ISBN 978-0387946207.
- Krug, J.; Spohn, H. (1988). "Universality classes for deterministic surface growth". *Physical Review A* **38** (8): 4271–4283. doi:10.1103/PhysRevA.38.4271. PMID 9900880.
- Land, Mark; Belew, Richard (1995). "No perfect two-state cellular automata for density classification exists". *Physical Review Letters* **74** (25): 1548–1550. Bibcode 1995PhRvL..74.5148L. doi:10.1103/PhysRevLett.74.5148. PMID 10058695.
- Li, Wentian (1987). "Power spectra of regular languages and cellular automata" (http://www.nslij-genetics.org/wli/pub/cs87-no-figure.pdf). *Complex Systems* **1**: 107–130.
- Li, Wentian (1992). "Phenomenology of nonlocal cellular automata". *Journal of Statistical Physics* **68** (5–6): 829–882. doi:10.1007/BF01048877.
- Moreira, Andres (2003). "Universality and decidability of number-conserving cellular automata". *Theoretical Computer Science* **292** (3): 711–721. arXiv:nlin.CG/0306032. doi:10.1016/S0304-3975(02)00065-8.
- Nagel, Kai (1996). "Particle hopping models and traffic flow theory". *Physical Review E* **53** (5): 4655–4672. doi:10.1103/PhysRevE.53.4655.
- Nagel, Kai; Schreckenberg, Michael (1992). "A cellular automaton model for freeway traffic". *Journal de Physique I* **2** (12): 2221–2229. doi:10.1051/jp1:1992277.
- Pivato, M. (2007). "Defect particle kinematics in one-dimensional cellular automata". *Theoretical Computer Science* **377** (1–3): 205–228. arXiv:math.DS/0506417. doi:10.1016/j.tcs.2007.03.014.
- Sukumar, N. (1998). *Effect of boundary conditions on cellular automata that classify density*. arXiv:comp-gas/9804001.
- Tadaki, Shin-ichi; Kikuchi, Macato (1994). "Jam phases in a two-dimensional cellular automaton model of traffic flow". *Physical Review E* **50** (6): 4564–4570. doi:10.1103/PhysRevE.50.4564.
- Wang, Bing-Hong; Kwong, Yvonne-Roamy; Hui, Pak-Ming (1998). "Statistical mechanical approach to Fukui-Ishibashi traffic flow models". *Physical Review E* **57** (3): 2568–2573. doi:10.1103/PhysRevE.57.2568.

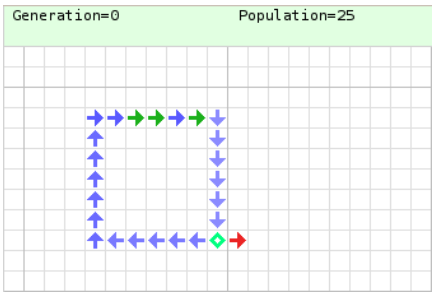
External links

- Rule 184 in Wolfram's atlas of cellular automata (http://atlas.wolfram.com/01/01/184/)

Von Neumann cellular automaton

Von Neumann cellular automata are the original expression of cellular automata, the development of which were prompted by suggestions made to John von Neumann by his close friend and fellow mathematician Stanisław Ulam. Their original purpose was to provide insight into the logical requirements for machine self-replication and were used in von Neumann's universal constructor.

Nobili's cellular automaton is a variation of von Neumann's cellular automaton, augmented with the ability for confluent cells to cross signals and store information. The former requires an extra three states, hence Nobili's cellular automaton has 32 states, rather than 29. Hutton's cellular automaton is yet another variation, which allows a loop of data, analogous to Langton's loops, to replicate.



A simple configuration in von Neumann's cellular automaton. A binary signal is passed repeatedly around the blue wire loop, using excited and quiescent *ordinary transmission states*. A confluent cell duplicates the signal onto a length of red wire consisting of *special transmission states*. The signal passes down this wire and constructs a new cell at the end. This particular signal (1011) codes for an east-directed special transmission state, thus extending the red wire by one cell each time. During construction, the new cell passes through several sensitised states, directed by the binary sequence.

Definition

Configuration

In general, cellular automata (CA) constitute an arrangement of **finite state automata** (FSA) that sit in positional relationships between one-another, each FSA exchanging information with those other FSAs to which it is positionally adjacent. In von Neumann's cellular automaton, the finite state machines (or ***cells***) are arranged in a two-dimensional Cartesian grid, and interface with the surrounding four cells. As von Neumann's cellular automaton was the first example to use this arrangement, it is known as the von Neumann neighbourhood.

The set of FSAs define a **cell space** of infinite size. All FSAs are identical in terms of state-transition function, or rule-set.

The **neighborhood** (a grouping function) is part of the state-transition function, and defines for any cell the set of other cells upon which the state of that cell depends.

All cells transition state synchronously, in step with a universal "clock" as in a synchronous digital circuit.

States

Each FSA of the von Neumann cell space can accept any of the 29 states of the rule-set. The rule-set is grouped into five orthogonal subsets. Each state includes the colour of the cell in the cellular automata program Golly [44] in (red, green, blue). They are

1. a **ground** state **U** (48, 48, 48)
2. the **transition** or **sensitised** states (in 8 substates)
    1. **S** (newly sensitised) (255, 0, 0)
    2. $\mathbf{S_0}$ - (sensitised, having received no input for one cycle) (255, 125, 0)
    3. $\mathbf{S_{00}}$ - (sensitised, having received no input for two cycles) (255, 175, 50)
    4. $\mathbf{S_{01}}$ - (sensitised, having received no input for one cycle and then an input for one cycle) (255, 200, 75)
    5. $\mathbf{S_{000}}$ - (sensitised, having received no input for three cycles) (251, 255, 0)
    6. $\mathbf{S_1}$ - (sensitised, having received an input for one cycle) (255, 150, 25)
    7. $\mathbf{S_{10}}$ - (sensitised, having received an input for one cycle and then no input for one cycle) (255, 255, 100)
    8. $\mathbf{S_{11}}$ - (sensitised, having received input for two cycles) (255, 250, 125)

3. the **confluent** states (in 4 states of excitation)
    1. $\mathbf{C_{00}}$ - quiescent (and will also be quiescent next cycle) (0, 255, 128)
    2. $\mathbf{C_{10}}$ - excited (but will be quiescent next cycle) (255, 255, 128)
    3. $\mathbf{C_{01}}$ - next-excited (now quiescent, but will be excited next cycle) (33, 215, 215)
    4. $\mathbf{C_{11}}$ - excited next-excited (currently excited and will be excited next cycle) (255, 128, 64)
4. the **ordinary transmission** states (in 4 directions, excited or quiescent, making 8 states)
    1. North-directed (excited and quiescent) (36, 200, 36) (106, 106, 255)
    2. South-directed (excited and quiescent) (106, 255, 106) (139, 139, 255)
    3. West-directed (excited and quiescent) (73, 255, 73) (122, 122, 255)
    4. East-directed (excited and quiescent) (27, 176, 27) (89, 89, 255)
5. the **special transmission** states (in 4 directions, excited or quiescent, making 8 states)
    1. North-directed (excited and quiescent) (191, 73, 255) (255, 56, 56)
    2. South-directed (excited and quiescent) (203, 106, 255) (255, 89, 89)
    3. West-directed (excited and quiescent) (197, 89, 255) (255, 73, 73)
    4. East-directed (excited and quiescent) (185, 56, 255) (235, 36, 36)

"Excited" states carry data, at the rate of one bit per state transition step.

Note that confluent states have the property of a one-cycle delay, thus effectively holding two bits of data at any given time.

Transmission state rules

The flow of bits between cells is indicated by the direction property. The following rules apply:

- Transmission states apply the OR operator to inputs, meaning a cell in a transmission state (ordinary or special) will be excited at time **t+1** if *any* of the inputs pointing to it is excited at time **t**
- Data passes from cells in the ordinary transmission states to other cells in the ordinary transmission states, according to the direction property (in the cell's direction only)
- Data passes from cells in the special transmission states to other cells in the special transmission states, according to the direction property (in the cell's direction only)
- The two subsets of transmission states, ordinary and special, are mutually antagonistic:
    - Given a cell **A** at time **t** in the excited ordinary transmission state
    - pointing to a cell **B** in any special transmission state
    - at time **t+1** cell **B** will become the ground state. The special transmission cell has been "destroyed."
    - a similar sequence will occur in the case of a cell in the special transmission state "pointing" to a cell in the ordinary transmission state

Confluent state rules

The following specific rules apply to confluent states:

- Confluent states do not pass data between each other.
- Confluent states take input from one or more ordinary transmission states, and deliver output to transmission states, ordinary and special, that are not directed toward the confluent state.
- Data are not transmitted against the transmission state direction property.
- Data held by a confluent state is lost if that state has no adjacent transmission state that is also not pointed at the confluent state.
- Thus, confluent-state cells are used as "bridges" from transmission lines of ordinary- to special-transmission state cells.

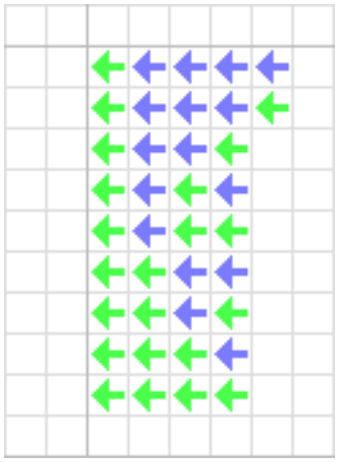
- The confluent state applies the AND operator to inputs, only "saving" an excited input if all potential inputs are excited simultaneously.
- Confluent cells delay signals by one generation more than OTS cells; this is necessary due to parity constraints.

Construction rules

Initially, much of the cell-space, the universe of the cellular automaton, is "blank," consisting of cells in the ground state **U**. When given an input excitation from a neighboring ordinary- or special transmission state, the cell in the ground state becomes "sensitised," transitioning through a series of states before finally "resting" at a quiescent transmission or confluent state.

The choice of which destination state the cell will reach is determined by the sequence of input signals. Therefore, the transition/sensitised states can be thought of as the nodes of a bifurcation tree leading from the ground-state to each of the quiescent transmission and confluent states.

In the following tree, the sequence of inputs is shown as a binary string after each step:



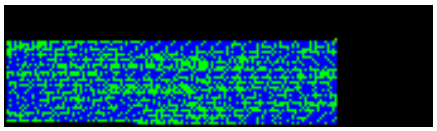
The nine cell types that can be constructed in von Neumann's CA. Here, binary signals pass along nine ordinary transmission lines, constructing a new cell when they encounter a ground state at the end. For example, the binary string 1011 is shown on the fifth line, and constructs the east-directed special transmission state - this is the same process as used in the automaton at the top of this page. Note that there is no interaction between neighbouring wires, unlike in Wireworld for example, allowing for a compact packing of components.

- a cell in the ground state **U**, given an input, will transition to the **S** (newly sensitised) state in the next cycle (1)
- a cell in the **S** state, given no input, will transition into the $\mathbf{S_0}$ state (10)
  - a cell in the $\mathbf{S_0}$ state, given no input, will transition into the $\mathbf{S_{00}}$ state (100)
    - a cell in the $\mathbf{S_{00}}$ state, given no input, will transition into the $\mathbf{S_{000}}$ state (1000)
      - a cell in the $\mathbf{S_{000}}$ state, given no input, will transition into the east-directed ordinary transmission state (10000)
      - a cell in the $\mathbf{S_{000}}$ state, given an input, will transition into the north-directed ordinary transmission state (10001)
    - a cell in the $\mathbf{S_{00}}$ state, given an input, will transition into the west-directed ordinary transmission state (1001)
  - a cell in the $\mathbf{S_0}$ state, given an input, will transition into the $\mathbf{S_{01}}$ state (101)
    - a cell in the $\mathbf{S_{01}}$ state, given no input, will transition into the south-directed ordinary transmission state (1010)
    - a cell in the $\mathbf{S_{01}}$ state, given an input, will transition into the east-directed special transmission state (1011)
- a cell in the **S** state, given an input, will transition into the $\mathbf{S_1}$ state (11)
  - a cell in the $\mathbf{S_1}$ state, given no input, will transition into the $\mathbf{S_{10}}$ state (110)
    - a cell in the $\mathbf{S_{10}}$ state, given no input, will transition into the north-directed special transmission state (1100)
    - a cell in the $\mathbf{S_{10}}$ state, given an input, will transition into the west-directed special transmission state (1101)
  - a cell in the $\mathbf{S_1}$ state, given an input, will transition into the $\mathbf{S_{11}}$ state (111)
    - a cell in the $\mathbf{S_{11}}$ state, given no input, will transition into the south-directed special transmission state (1110)
    - a cell in the $\mathbf{S_{11}}$ state, given an input, will transition into the quiescent confluent state $\mathbf{C_{00}}$ (1111)

Note that:

- one more cycle of input (four after the initial sensitization) is required to build the east- or north-directed ordinary transmission state than any of the other states (which require three cycles of input after the initial sensitization),
- the "default" quiescent state resulting in construction is the east-directed ordinary transmission state- which requires an initial sensitization input, and then four cycles of no input.

Destruction rules



Roughly 4000 bits of data in a curled up "tape" constructing a complex pattern. This uses a 32-state variation of von Neumann cellular automata known as Hutton32.

- An input into a confluent-state cell from a special-transmission state cell will result in the confluent state cell being reduced back to the ground state.
- Likewise, an input into an ordinary transmission-state cell from a special-transmission state cell will result in the ordinary-transmission state cell being reduced back to the ground state.
- Conversely, an input into a special transmission-state cell from an ordinary-transmission state cell will result in the special-transmission state cell being reduced back to the ground state.

References

- Von Neumann, J. and A. W. Burks (1966). Theory of self-reproducing automata. Urbana,, University of Illinois Press. [1]

External links

- Golly [2] - supports von Neumann's CA along with the Game of Life, and other rulesets.

References

[1] http://web.archive.org/web/20080613231120/http://www.walenz.org/vonNeumann/index.html

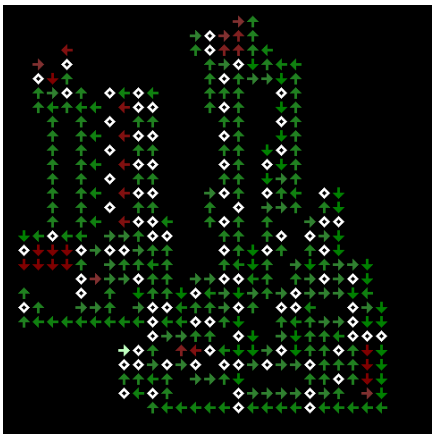
[2] http://golly.sourceforge.net

Nobili cellular automata

Nobili cellular automata are a variation of von Neumann cellular automata, in which additional states provide means of memory and the interference-free crossing of signal. Nobili cellular automata are the invention of Renato Nobili, a professor of physics at the University of Padova in Padova, Italy. Von Neumann specifically excluded the use of states dedicated to the crossing of signal.

The confluent state is altered, so that it acts as a signal crossing organ if exactly two signal paths are incident (they enter and leave the confluent state), or acts as a memory organ if only inputs exist.

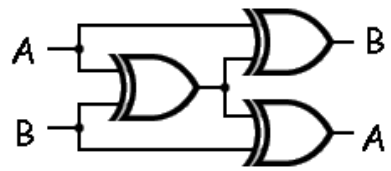
The benefit of these alterations to the state set of von Neumann cellular automata is that signal crossing is greatly eased, configurations are slightly smaller than the corresponding configuration of von Neumann cellular automata, and the computational throughput is increased.



λ_G, a minimal self-replicating configuration in Nobili cellular automata

Signal crossing in vNCA

In von Neumann's original cellular automaton, the crossing of signals is much more difficult. The most widely used signal crossing organs are the *coded channel* (devised by von Neumann himself), Gorman's *real-time crossing organ*, and the *Mukhopadhyay crossing organ*. The coded channel can only cross individual pulses; the others are capable of crossing entire packets without interference, analogous the crossing organ in Nobili's cellular automaton. The Mukhopadhyay crossing organ comprises three XOR gates, in the arrangement shown (left).



Schematic of the Mukhopadhyay crossing organ, showing the three exclusive-or gates.

Signal crossing in NCA

In Nobili cellular automaton, a signal crossing organ consists of a single confluent cell, with two perpendicular input paths and two perpendicular output paths. Due to the substantially reduced size (as compared with any of the vNCA crossing organs), self-replicating machines are much more compact in NCA. For example, the smallest replicator so far, λ_G, comprises only 485 somatic cells.

Memory storage in vNCA

Storing memory in vNCA can be done in multiple ways. One of these (the electronic method) is to create a loop of OTS cells with an excited pulse travelling around it. By far the most common way (the electro-mechanical method) is to use a special transmission state to construct and delete an ordinary transmission state, to act as a gate. Slight modifications can yield a plethora of different gates, including latches, pulse dividers, and one-time gates.

Memory storage in NCA

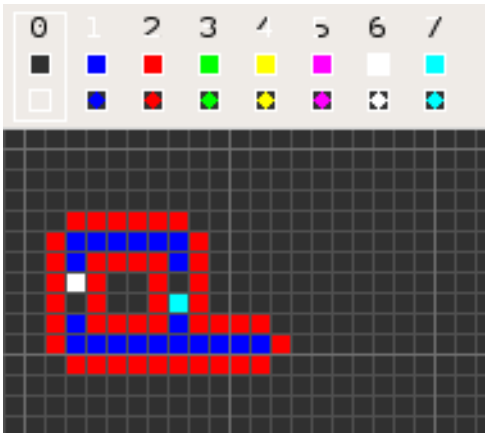
In Nobili's cellular automaton, this task is also simplified. A confluent cell with no outputs 'holds' a pulse of excitation until an output is created. In the diagram of λ_G (above, right), the excited confluent cell is displayed in orange. It will remain in this state until an adjacent OTS cell is created, at which point the information will flow into the next confluent cell.

References

- Buckley, W. R. *Signal Crossing Solutions in von Neumann self-replicating cellular automata* (Automata 2008).

Codd's cellular automaton

Codd's cellular automaton is a cellular automaton (CA) devised by the British computer scientist Edgar F. Codd in 1968. It was designed to recreate the computation- and construction-universality of von Neumann's CA but with fewer states: 8 instead of 29. Codd showed that it was possible to make a self-reproducing machine in his CA, in a similar way to von Neumann's universal constructor but never gave a complete implementation.



A simple configuration in Codd's cellular automaton. Signals pass along wire made of cells in state 1 (blue) sheathed by cells in state 2 (red). Two signal trains circulate around a loop and are duplicated at a T-junction onto an open-ended section of wire. The first (7-0) causes the sheathed end of the wire to become exposed. The second (6-0) re-sheathes the exposed end, leaving the wire one cell longer than before.

Historical Context

In the 1940s and 50's, John von Neumann posed the following problem[1] :

- What kind of logical organization is sufficient for an automaton to be able to reproduce itself?

He was able to construct a Universal Constructor with a square grid and 29 states. E.F. Codd found a simpler machine with only eight states[2] . Therefore von Neumann's question had to be modified:

- What kind of logical organization is *necessary* for an automaton to be able to reproduce itself?

Three years after Codd's work, Edwin Roger Banks showed a 4-state CA (appendix IV in his thesis) that was also computation- and construction-universal but again didn't implement a self-reproducing machine[3] .

John Devore, in his 1973 master's thesis, was able to greatly reduce the size of Codd's machine design while using almost the same 8-state CA. More than just a theoretical device, Devore's machine was possibly the first implementation of a self-reproducing CA. The machine itself could fit inside computer memory but the data tape stretched for thousands of cells, so simulation of the entire self-reproduction cycle was not possible. Decades later,

Devore's original design would complete self-reproduction in simulation using the Golly program.

In 1984, Christopher Langton extended Codd's cellular automaton to create Langton's loops[4] , which also exhibits reproductive behavior but uses only a very small number of cells compared to the hundreds of thousands needed for self-reproduction in von Neumann, Codd or Banks' cellular automata.

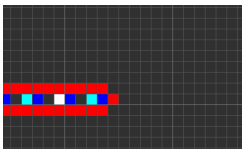
Comparison of CA rulesets

CA	number of states	symmetries	computation- and construction-universal	size of self-reproducing machine
von Neumann	29	none	yes	130,622 cells
Codd	8	rotations	yes	283,126,588 cells[5]
Devore	8	rotations	yes	94,794 cells
Banks-IV	4	rotations and reflections	yes	unknown
Langton's loops	8	rotations	no	86 cells

Specification

Codd's CA has 8-states and the von Neumann neighborhood with rotational symmetry.

The table below shows the signal-trains needed to accomplish different tasks. Some of the signal trains need to be separated by two blanks (state 1) on the wire to avoid interference, so the 'extend' signal-train used in the image at the top appears here as '70116011'.



The construction arm in Codd's CA can be moved into position using the commands listed in the text. Here the arm turns left, then right, then writes a cell before retracting along the same path.

purpose	signal train
extend	70116011
extend_left	4011401150116011
extend_right	5011501140116011
retract	4011501160116011
retract_left	5011601160116011
retract_right	4011601160116011
mark	701160114011501170116011
erase	601170114011501160116011
sense	70117011
cap	40116011
inject_sheath	701150116011
inject_trigger	60117011701160116011

Universal computer-constructor

Codd designed a self-replicating computer in the cellular automaton, based on Wang's W-machine. However, the design was so colossal that it evaded implementation until 2009, when Tim Hutton constructed an explicit configuration[5] . There were some minor errors in Codd's design, so Hutton's implementation differs slightly, in both the configuration and the ruleset.

References

[1] von Neumann, John; Burks, Arthur W. (1966). "*Theory of Self-Reproducing Automata.*" (http://web.archive.org/web/20080105213853/http://www.walenz.org/vonNeumann/index.html) (Scanned book online). www.walenz.org. Archived from the original (http://www.walenz.org/vonNeumann/index.html) on 2008-01-05. . Retrieved 2008-02-29.

[2] Codd, Edgar F. (1968). *Cellular Automata*. Academic Press, New York.

[3] Banks, Edwin (1971). *Information Processing and Transmission in Cellular Automata* (http://www.bottomlayer.com/bottom/banks/banks_commentary.htm). PhD thesis, MIT, Department of Mechanical Engineering. .

[4] Langton, C. G. (1984). "Self-Reproduction in Cellular Automata". *Physica D: Nonlinear Phenomena* **10** (1-2): 135–144. doi:10.1016/0167-2789(84)90256-2.

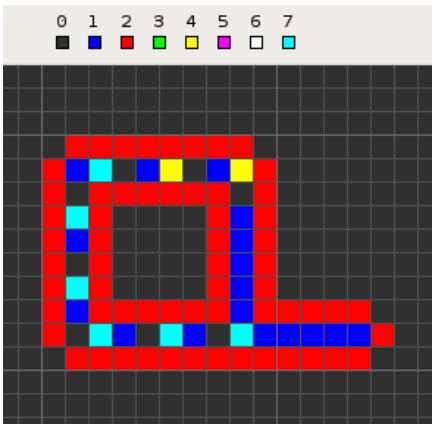
[5] Hutton, Tim J. (2010). "Codd's self-replicating computer" (http://www.sq3.org.uk/papers/Hutton_CoddsSelfReplicatingComputer_2010.pdf). *Artificial Life* **16** (2): 99–117. doi:10.1162/artl.2010.16.2.16200. PMID 20067401. .

External links

- The Rule Table Repository (http://code.google.com/p/ruletablerepository/) has the transition table for Codd's CA.
- Golly (http://golly.sourceforge.net) - supports Codd's CA along with the Game of Life, and other rulesets.
- Download the complete machine (13MB) (http://code.google.com/p/ruletablerepository/wiki/CoddsDesign) and more details.

Langton's loops

Langton's loops are a particular "species" of artificial life in a cellular automaton created in 1984 by Christopher Langton. They consist of a loop of cells containing genetic information, which flows continuously around the loop and out along an "arm" (or pseudopod), which will become the daughter loop. The "genes" instruct it to make three left turns, completing the loop, which then disconnects from its parent.



Langton's Loop, in the starting configuration.

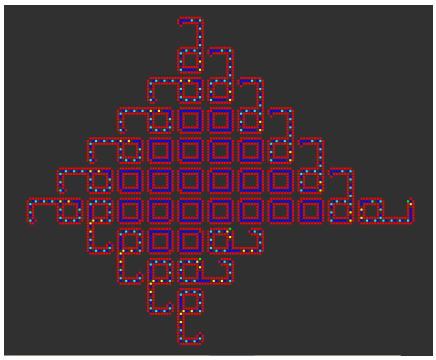
History

In 1952 John von Neumann created the first cellular automaton (CA) with the goal of creating a self-replicating machine.[1] This automaton was necessarily very complex due to its computation- and construction-universality. In 1968 Edgar F. Codd reduced the number of states from 29 in von Neumann's CA to 8 in his.[2] When Christopher Langton did away with the universality condition, he was able to significantly reduce the automaton's complexity. Its self-replicating loops are based on one of the simplest elements in Codd's automaton, the periodic emitter.

Specification

Langton's Loops run in a CA that has 8 states, and uses the von Neumann neighborhood with rotational symmetry. The transition table can be found here: [3].

As with Codd's CA, Langton's Loops consist of sheathed wires. The signals travel passively along the wires until they reach the open ends, when the command they carry is executed.


A colony of loops. The ones in the centre are "dead".

Colonies

Because of a particular property of the loops' "pseudopodia", they are unable to reproduce into the space occupied by another loop. Thus, once a loop is surrounded, it is incapable of reproducing, resulting in a coral-like colony with a thin layer of reproducing organisms surrounding a core of inactive "dead" organisms. Unless provided unbounded space, the colony's size will be limited. The maximum population will be asymptotic to $\left\lfloor \frac{A}{121} \right\rfloor$, where A is the total area of the space in cells.

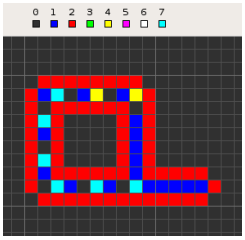
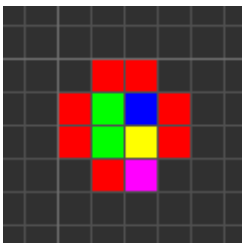
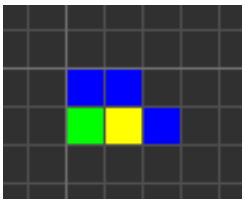
Encoding of the genome

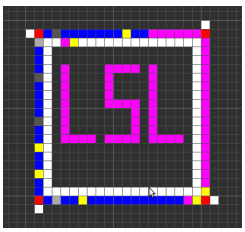
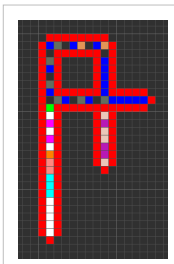
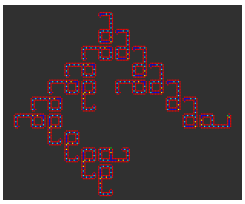
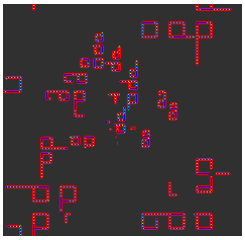
The loops' genetic code is stored as a series of nonzero-zero state pairs. The standard loop's genome is illustrated in the picture at the top, and may be stated as a series of numbered states starting from the T-junction and running clockwise: 70-70-70-70-70-70-40-40. The '70' command advances the end of the wire by one cell, while the '40-40' sequence causes the left turn. State 3 is used as a temporary marker for several stages.

While the roles of states 0,1,2,3,4 and 7 are similar to Codd's CA, the remaining states 5 and 6 are used instead to mediate the loop replication process. After the loop has completed, state 5 travels counter-clockwise along the sheath of the parent loop to the next corner, causing the next arm to be produced in a different direction. State 6 temporarily joins the genome of the daughter loop and initialises the growing arm at the next corner it reaches.

The genome is used a total of six times: once to extend the pseudopod to the desired location, four times to complete the loop, and again to transfer the genome into the daughter loop. Clearly, this is dependent on the fourfold rotational symmetry of the loop; without it, the loop would be incapable of containing the information required to describe it. The same use of symmetry for genome compression is used in many biological viruses, such as the icosahedral adenovirus.

Comparison of related CA loops

CA	number of states	neighborhood	number of cells (typical)	replication period (typical)	thumbnail
Langton's loops[4] (1984): The original self-reproducing loop.	8	von Neumann	86	151	
Byl's loop[5] (1989): By removing the inner sheath, Byl reduced the size of the loop.	6	von Neumann	12	25	
Chou-Reggia loop[6] (1993): A further reduction of the loop by removing all sheaths.	8	von Neumann	5	15	

Tempesti loop[7] (1995): Tempesti added construction capabilities to his loop, allowing patterns to be written inside the loop after reproduction.	10	Moore	148	304	
Perrier loop[8] (1996): Perrier added a program stack and an extensible data tape to Langton's loop, allowing it to compute anything computable.	64	von Neumann	158	235	
SDSR loop[9] (1998): With an extra structure-dissolving state added to Langton's loops, the SDSR loop has a limited lifetime and dissolves at the end of its life cycle. This allows continuous growth and turn-over of generations.	9	von Neumann	86	151	
Evoloop[10] (1999): An extension of the SDSR loop, Evoloop is capable of interaction with neighboring loops as well as of evolution. Often, the greatest selection pressure in a colony of Evoloops is the competition for space, and natural selection favors the smallest functional loop present. Further studies demonstrated more complexity than originally thought in the Evoloop system.[11]	9	von Neumann	149	363	
Sexyloop[12] (2007): Sexyloop is a modification of Evoloop in which collisions often result in transfer of genetic material between loops. This increases diversity in evolution of new species of loops.	10	von Neumann	?	?	

References

[1] von Neumann, John; Burks, Arthur W. (1966). "*Theory of Self-Reproducing Automata.*" (http://web.archive.org/web/20080105213853/http://www.walenz.org/vonNeumann/index.html) (Scanned book online). www.walenz.org. Archived from the original (http://www.walenz.org/vonNeumann/index.html) on 2008-01-05. . Retrieved 2008-02-29.

[2] Codd, Edgar F. (1968). *"Cellular Automata"*. Academic Press, New York.

[3] http://code.google.com/p/ruletablerepository/

[4] C. G. Langton (1984). "Self-reproduction in cellular automata". *Physica D* **10**: 135–144. doi:10.1016/0167-2789(84)90256-2.

[5] J. Byl (1989). "Self-Reproduction in small cellular automata". *Physica D* **34**: 295–299. doi:10.1016/0167-2789(89)90242-X.

[6] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng (1993). "Simple systems that exhibit self-directed replication" (http://www.sciencemag.org/content/259/5099/1282.short). *Science* **259** (5099): 1282–1287. doi:10.1126/science.259.5099.1282. PMID 17732248. .

[7] G. Tempesti (1995). "A New Self-Reproducing Cellular Automaton Capable of Construction and Computation" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.7578). *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life*. Granada, Spain: Lecture Notes in Artificial Intelligence, 929, Springer Verlag, Berlin. pp. 555–563. .

[8] J.-Y. Perrier, M. Sipper, and J. Zahnd (1996). "Toward a viable, self-reproducing universal computer" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.3200). *Physica D* **97**: 335–352. doi:10.1016/0167-2789(96)00091-7. .

[9] Hiroki Sayama (1998). "Introduction of Structural Dissolution into Langton's Self-Reproducing Loop". *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*. Los Angeles, California: MIT Press. pp. 114–122.

[10] Hiroki Sayama (1999). "Toward the Realization of an Evolving Ecosystem on Cellular Automata" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.391). *Proceedings of the Fourth International Symposium on Artificial Life and Robotics (AROB 4th '99)*. Beppu, Oita, Japan. pp. 254–257. .

[11] Chris Salzberg and Hiroki Sayama (2004). "Complex genetic evolution of artificial self-replicators in cellular automata" (http://www3.interscience.wiley.com/journal/109860047/abstract). *Complexity* **10**: 33–39. doi:10.1002/cplx.20060. .

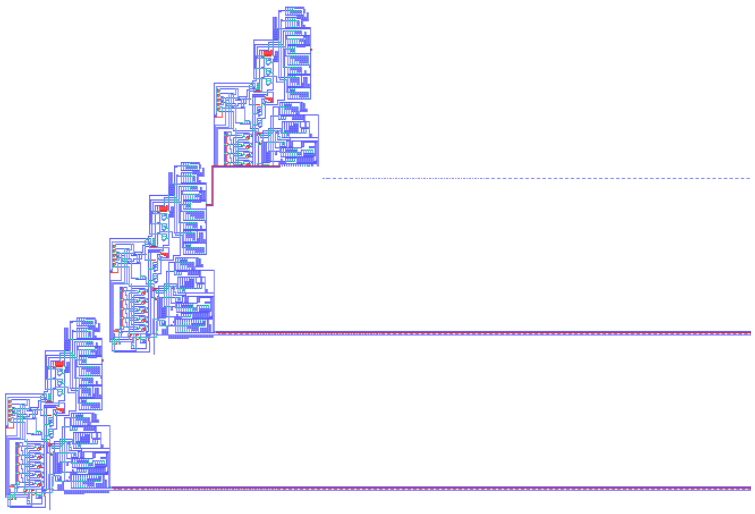
[12] Nicolas Oros and C. L. Nehaniv (2007). "Sexyloop: Self-Reproduction, Evolution and Sex in Cellular Automata" (https://uhra.herts.ac.uk/dspace/bitstream/2299/1735/1/901918.pdf). *The First IEEE Symposium on Artificial Life (April 1-5, 2007, Hawaii, USA)*. pp. 130–138. .

External links

- visual representation (http://necsi.edu/postdocs/sayama/sdsr/java/) of several of the self-replicating loops in a Java applet
- The Rule Table Repository (http://code.google.com/p/ruletablerepository/) has the transition tables for many of the CA mentioned above.
- Golly (http://golly.sourceforge.net) - supports Langton's Loops along with the Game of Life, and other rulesets.

Von Neumann universal constructor

John von Neumann's **Universal Constructor** is a self-replicating machine in a cellular automata (CA) environment. It was designed in the 1940s, without the use of a computer. The fundamental details of the machine were published in von Neumann's book *Theory of Self-Reproducing Automata*, completed in 1966 by Arthur W. Burks after von Neumann's death.[2]



The first implementation of von Neumann's self-reproducing universal constructor.[1] Three generations of machine are shown, the second has nearly finished constructing the third. The lines running to the right are the tapes of genetic instructions, which are copied along with the body of the machines. The machine shown runs in a 32-state version of von Neumann's cellular automata environment, not his original 29-state specification.

Von Neumann's specification defined the machine as using 29 states, these states constituting means of signal carriage and logical operation, and acting upon signals represented as bit streams. A 'tape' of cells encodes the sequence of actions to be performed by the machine. Using a writing head (termed a construction arm) the machine can print out (construct) a new pattern of cells, allowing it to make a complete copy of itself, and the tape.

Purpose

Von Neumann's design has traditionally been understood to be a demonstration of the logical requirements for machine self-replication.[3] However it is clear that far simpler machines can achieve self-replication. Examples

include trivial crystal-like growth, template replication and Langton's loops. But von Neumann was interested in something more profound: construction universality and evolution.[4]

This universal constructor can be seen as an abstract simulation of a physical universal assembler.

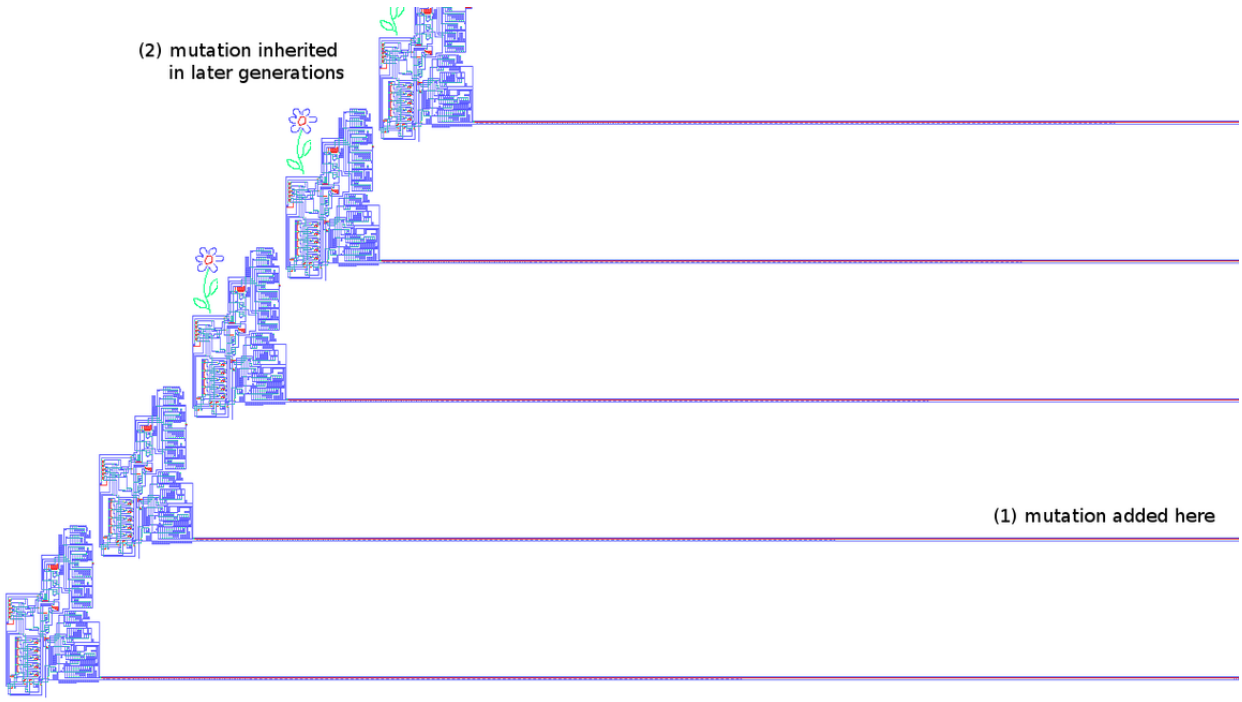
Note that the simpler self-replicating CA structures (especially, Byl's loop and the Chou-Reggia loop) cannot exist in a wide variety of forms and thus have very limited evolvability. Other CA structures such as the Evoloop are somewhat evolvable but still don't support open-ended evolution. Commonly, simple replicators do not fully contain the machinery of construction, there being a degree to which the replicator is information copied by its surrounding environment. Although the Von Neumann design is a logical construction, it is in principle a design that could be instantiated as a physical machine. The issue of the environmental contribution to replication is somewhat open, since there are different conceptions of raw material and its availability.

The concept of a *universal constructor* is non-trivial because of the existence of garden of eden patterns. But a simple definition is that a universal constructor is able to construct any finite pattern of non-excited (quiescent) cells.

Von Neumann's crucial insight is that part of the replicator has a double use; being both an active component of the construction mechanism, and being the target of a passive copying process. This part is played by the tape of instructions in Von Neumann's combination of universal constructor plus instruction tape.

The combination of a universal constructor and a tape of instructions would i) allow self-replication, and also ii) guarantee that the open-ended complexity growth observed in biological organisms was possible.[3] The image below illustrates this possibility.

This insight is all the more remarkable because it preceded the discovery of the structure of the DNA molecule by Watson and Crick, though it followed the Avery-MacLeod-McCarty experiment which identified DNA as the molecular carrier of genetic information in living organisms.[5] The DNA molecule is processed by separate mechanisms that carry out its instructions and copy the DNA for insertion for the newly constructed cell. The ability to achieve open-ended evolution lies in the fact that, just as in nature, errors (mutations) in the copying of the genetic tape can lead to viable variants of the automaton, which can then evolve via natural selection.



A demonstration of the ability of von Neumann's machine to support inheritable mutations. (1) At an earlier timestep, a mutation was manually added to the second generation machine's tape. (2) Later generations both display the phenotype of the mutation (a drawing of a flower) and pass the mutation on to their children, since the tape is copied each time. This example illustrates how von Neumann's design allows for complexity growth (in theory) since the tape could specify a machine that is more complex than the one making it.

Implementation

Arthur Burks and others extended the work of von Neumann, giving a much clearer and complete set of details regarding the design and operation of von Neumann's self-replicator. The work of J. W. Thatcher is particularly noteworthy, for he greatly simplified the design. Still, their work did not yield a complete design, cell by cell, of a configuration capable of demonstrating self-replication.

Renato Nobili and Umberto Pesavento published the first fully implemented self-reproducing cellular automaton in 1995, nearly fifty years after von Neumann's work.[1] [6] They used a 32-state cellular automaton instead of von Neumann's original 29-state specification, extending it to allow for easier signal-crossing and a more compact design. They also published an implementation of a general constructor within the original 29-state CA but not one capable of complete replication - the configuration cannot duplicate its tape, nor can it trigger its offspring; the configuration can only construct.[6] [7]

In 2007, Nobili published a 32-state implementation that uses run-length encoding to greatly reduce the size of the tape. [8]

In 2008, William R. Buckley published two configurations which are self-replicators within the original 29-state CA of von Neumann.[7] Buckley claims that the crossing of signal within von Neumann 29-state cellular automata is not necessary to the construction of self-replicators.[7] Buckley also points out that for the purposes of evolution, each replicator should return to its original configuration after replicating, in order to be capable (in theory) of making more than one copy. As published, the 1995 design of Nobili-Pesavento does not fulfill this requirement but the 2007 design of Nobili does; the same is true of Buckley's configurations.

In 2004, D. Mange et al. reported an implementation of a self-replicator that is consistent with the designs of von Neumann.[9]

In 2009, Buckley published with Golly a third configuration for von Neumann 29-state cellular automata, which can perform either holistic self-replication, or self-replication by partial construction. This configuration also demonstrates that signal crossing is not necessary to the construction of self-replicators within von Neumann 29-state cellular automata.[10]

C. L. Nehaniv in 2002, and also Y. Takada et al. in 2004, proposed a universal constructor directly implemented upon an asynchronous cellular automaton, rather than upon a synchronous cellular automaton. [11] [12]

Comparison of implementations

implementation	source	ruleset	rectangular area	number of cells	length of tape	ratio	timesteps for replication	tape code compression	tape code length	replication mechanism	replication type	growth rate
Nobili-Pesavento, 1995 [1]	[13]	Nobili 32-state	97 × 170	6,329	145,315	22.96	6.34×10^{10}	none	5 bits	holistic constructor	non-repeatable	linear
Nobili, 2007	SR_CCN_AP.EVN [8][14]	Nobili 32-state	97 × 100	5,313	56,325	10.60	9.59×10^{9}	run-length limited encoding	5 bits	holistic constructor	repeatable	super-linear
Buckley, 2009	codon3.rle	Nobili 32-state	116 × 95	4,855	23,577	4.86	1.63×10^{9}	auto-retraction/bit generation/code overlay	3 bits	holistic constructor	repeatable	super-linear
Buckley, 2008	codon4.rle [10]	Nobili 32-state	109 × 59	3,574	37,780	10.57	4.31×10^{9}	auto-retraction/bit generation	4 bits	holistic constructor	repeatable	linear
Buckley, 2008	codon5.rle [10]	Nobili 32-state	112 × 50	3,343	44,155	13.21	5.87×10^{9}	auto-retraction	5 bits	holistic constructor	repeatable	linear
Buckley, 2008[7]	replicator.mc [15]	von Neumann 29-state	312 × 132	18,589	294,844	15.86	2.61×10^{11}	auto-retraction	5 bits	holistic constructor	repeatable	linear
Buckley, 2009	PartialReplicator.mc [10]	von Neumann 29-state	NA	NA	NA	-	~1.12×10^{14}	none	4 bits	partial constructor	repeatable	linear

It should be noted that none of the configurations discussed in this article is a universal constructor; none could, for instance, construct the real-time crossing organ devised by Gorman.[7] To date, no configuration capable of universal construction has been demonstrated for the 29-state model of von Neumann.

Practicality

Computational cost

All the implementations of von Neumann's self-reproducing machine require considerable resources to run on computer. For example, in the Nobili-Pesavento 32-state implementation shown above, while the body of the machine is just 6,329 non-empty cells (within a rectangle of size 97x170), it requires a tape that is 145,315 cells long, and takes 63 billion timesteps to replicate. A simulator running at 1,000 timesteps per second would take over 2 years to make the first copy. In 1995, when the first implementation was published, the authors had not seen their own machine replicate. However, in 2008, the hashlife algorithm was extended to support the 29-state and 32-state

rulesets in Golly [2]. On a modern desktop PC, replication now takes only a few minutes, although a significant amount of memory is required.

Evolvability

Von Neumann's stated problem was evolution [16]: how is the complexity growth and evolvability of biological organisms possible? His machine shows how it is logically possible, by using a universal constructor, but does not show how it is possible in practice. In his unfinished work he briefly considers conflict and interactions between replicators [17] but in practice his model is not likely to become a useful unit of evolution because it is too fragile.[3]

Animation gallery

References

[1] Pesavento, Umberto (1995), "An implementation of von Neumann's self-reproducing machine" (http://web.archive.org/web/20070418081628/http://dragonfly.tam.cornell.edu/~pesavent/pesavento_self_reproducing_machine.pdf) (PDF), *Artificial Life* (MIT Press) **2** (4): 337–354, doi:10.1162/artl.1995.2.337, PMID 8942052,

[2] von Neumann, John; Burks, Arthur W. (1966) (Scanned book online), Theory of Self-Reproducing Automata. (http://web.archive.org/web/20080105213853/www.walenz.org/vonNeumann/index.html), www.walenz.org, archived from the original (http://www.walenz.org/vonNeumann/index.html) on 2008-01-05, , retrieved 2008-02-29

[3] McMullin, B. (2000), "John von Neumann and the Evolutionary Growth of Complexity: Looking Backwards, Looking Forwards..." (http://www.eeng.dcu.ie/~alife/bmcm-alj-2000/), *Artificial Life* **6** (4): 347–361, doi:10.1162/106454600300103674, PMID 11348586,

[4] http://www.walenz.org/vonNeumann/page0110.html

[5] Rocha, L.M., "Von Neumann and Natural Selection." (http://informatics.indiana.edu/rocha/i-bic/pdfs/ibic_lecnotes_c6.pdf), *Lecture Notes of I-585-Biologically Inspired Computing Course, Indiana University*,

[6] Nobili, Renato; Pesavento, Umberto (1996), "Generalised von Neumann's Automata" (http://www.pd.infn.it/~rnobili/pdf_files/jvnconstr.pdf), in Besussi, E.; Cecchini, A. (PDF), *Proc. Artificial Worlds and Urban Studies, Conference 1*, Venice: DAEST,

[7] Buckley, William R. (2008), "Signal Crossing Solutions in von Neumann Self-replicating Cellular Automata", in Andrew Adamatzky, Ramon Alonso-Sanz, Anna Lawniczak, Genaro Juarez Martinez, Kenichi Morita, Thomas Worsch, *Proc. Automata 2008*, Luniver Press

[8] http://www.pd.infn.it/~rnobili/wjvn/index.htm

[9] Mange, Daniel; Stauffer, A.; Peparaolo, L.; Tempesti, G. (2004), "A Macroscopic View of Self-replication", *Proceedings of the IEEE* **92** (12): 1929–1945, doi:10.1109/JPROC.2004.837631

[10] http://www.sourceforge.net/projects/golly

[11] Nehaniv, Chrystopher L. (2002), "Self-Reproduction in Asynchronous Cellular Automata", *2002 NASA/DoD Conference on Evolvable Hardware (15-18 July 2002, Alexandria, Virginia, USA)*, IEEE Computer Society Press, pp. 201–209

[12] Takada, Yousuke; Isokawa, Teijiro; Peper, Ferdinand; Matsui, Nobuyuki (2004), "Universal Construction on Self-Timed Cellular Automata", in Sloot, P.M.A., *ACRI 2004, LNCS 3305*, pp. 21–30

[13] http://www.sq3.org.uk/Evolution/JvN/

[14] http://sourceforge.net/mailarchive/forum.php?thread_name=aac498730807310217udbc6fd8y809c16003ceb3782%40mail.gmail.com&forum_name=golly-test

[15] http://tomas.rokicki.com/golly-1.5a-win.zip

[16] http://www.walenz.org/vonNeumann/page0110.html

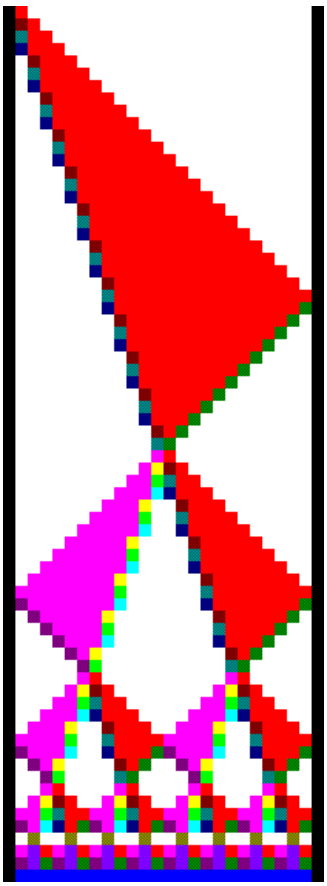
[17] http://www.walenz.org/vonNeumann/page0147.html

External links

- Golly - the Cellular Automata Simulation Accelerator (http://golly.sourceforge.net) Very fast implementation of state transition and support for JvN, GoL, Wolfram, and other systems.
- von Neumann's Self-Reproducing Universal Constructor (http://www.sq3.org.uk/Evolution/JvN/) The original Nobili-Pesavento source code, animations and Golly files of the replicators.
- John von Neumann's 29 state Cellular Automata Implemented in OpenLaszlo (http://www.donhopkins.com/drupal/node/41) by Don Hopkins
- A Catalogue of Self-Replicating Cellular Automata. (http://uncomp.uwe.ac.uk/automata2008/buckley/buckley.pdf) This catalogue complements the *Proc. Automata 2008* volume.

Firing squad synchronization problem

The **firing squad synchronization problem** is a problem in computer science and cellular automata in which the goal is to design a cellular automaton that, starting with a single active cell, eventually reaches a state in which all cells are simultaneously active. It was first proposed by John Myhill in 1957 and published (with a solution) in 1962 by Edward Moore.



One solution to the FSSP using 15 states and 3n units of time. Time increases from top to bottom.

Problem statement

The name of the problem comes from an analogy with real-world firing squads: the goal is to design a system of rules according to which a general can command a squad of troops to fire, and cause them to all fire their guns simultaneously.

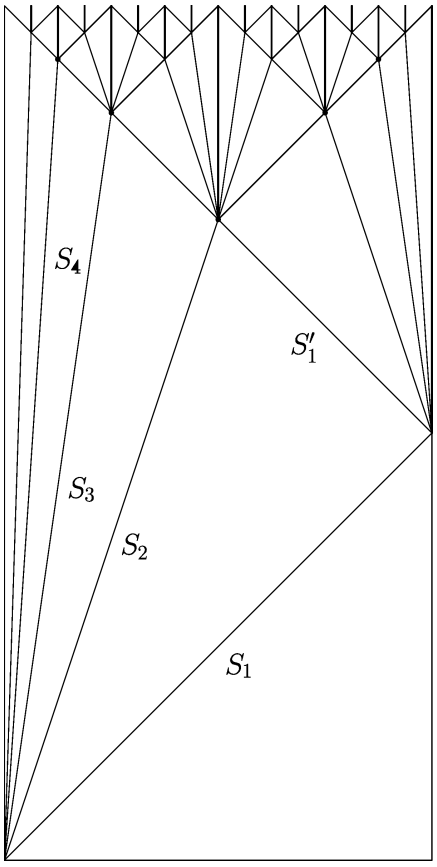
More formally, the problem concerns cellular automata, arrays of finite state machines called "cells" arranged in a line, such that at each time step each machine transitions to a new state as a function of its previous state and the states of its two neighbors in the line. For the firing squad problem, the line consists of a finite number of cells, and the rule according to which each machine transitions to the next state should be the same for all of the cells interior to the line, but the transition functions of the two endpoints of the line are allowed to differ, as these two cells are each missing a neighbor on one of their two sides.

The states of each cell include three distinguished states: "active", "quiescent", and "firing", and the transition function must be such that a cell that is quiescent and whose neighbors are quiescent remains quiescent. Initially, at time $t = 0$, all states are quiescent except for the cell at the far left (the general), which is active. The goal is to design a set of states and a transition function such that, no matter how long the line of cells is, there exists a time `t` such that every cell transitions to the firing state at time `t`, and such that no cell belongs to the firing state prior to time `t`.

Solutions

The first solution to the FSSP was found by John McCarthy and Marvin Minsky and was published in *Sequential Machines* by Moore. Their solution involves propagating two waves down the line of soldiers: a fast wave and a slow wave moving three times as slow. The fast wave bounces off the other end of the line and meets the slow wave in the centre. The two waves then split into four waves, a fast and slow wave moving in either direction from the centre, effectively splitting the line into two equal parts. This process continues, subdividing the line until each division is of length 1. At this moment, every soldier fires. This solution requires $3n$ units of time for n soldiers.

A solution using a minimal amount of time was later found by Eiichi Goto (1962); his solution used thousands of states, but required only $2n - 2$ units of time for n soldiers. A solution using a smaller amount of time cannot exist. Waksman (1966) improved this solution, using the same amount of time but only 16 states. Balzer (1967) further improved it to eight states, and proved that no four-state solution exists; Peter Sanders later found that Balzer's search procedure was incomplete, but managed to reaffirm the four-state non-existence result through a corrected search procedure. Finally, the best currently known solution, a minimal-time solution using six states, was introduced by Jacques Mazoyer (1987). It is still unknown whether a five-state solution exists.



A solution using 2n-2 units of time. Time increases from bottom to top.

In the minimal-time solutions, the general sends to the right signals $S_1, S_2, S_3, ..., S_i$ at speeds 1, 1/3, 1/7, ..., $1/(2^{i-1} - 1)$. The signal S_1 reflects at the right end of the line, and meets signal S_i (for $i \geq 2$) at cell $n/2^{i-1}$. When S_1 reflects, it also creates a new general at the right end. Signals S_i are constructed using auxiliary signals, which propagate to the left. Every second time a signal moves (to the right), it sends an auxiliary signal to the left. S_1 moves on its own at speed 1 while each of the slower signals moves only when it gets an auxiliary signal.

Generalizations

The firing squad synchronization problem has been generalized to many other types of cellular automaton, including higher dimensional arrays of cells (Shinahr 1974). Variants of the problem with different initial conditions have also been considered (Kobayashi & Goldstein 2005).

Solutions to the firing squad problem may also be adapted to other problems. For instance, Patrick Fischer (1965) designed a cellular automaton algorithm to generate the prime numbers based on an earlier solution to the firing squad synchronization problem.

References

- Balzer, Robert (1967), "An 8-state minimal time solution to the firing squad synchronization problem", *Information and Control* **10** (1): 22–42, doi:10.1016/S0019-9958(67)90032-0.
- Fischer, Patrick C. (1965), "Generation of primes by a one-dimensional real-time iterative array", *Journal of the ACM* **12** (3): 388–394, doi:10.1145/321281.321290.
- Goto, Eiichi (1962), *A minimal time solution of the firing squad problem*, Dittoed course notes for Applied Mathematics 298, Cambridge, MA: Harvard University, pp. 52-59. As cited by Waksman (1966).
- Kobayashi, Kojiro; Goldstein, Darin (2005), "On formulations of firing squad synchronization problems" [1], *Unconventional Computation*, Lecture Notes in Computer Science, **3699**, Springer-Verlag, pp. 157–168, doi:10.1007/11560319_15.
- Mazoyer, Jacques (1987), "A six-state minimal time solution to the firing squad synchronization problem", *Theoretical Computer Science* **50** (2): 183–238, doi:10.1016/0304-3975(87)90124-1.
- Moore, F. R.; Langdon, G. G. (1968), "A generalized firing squad problem", *Information and Control* **12** (3): 212–220, doi:10.1016/S0019-9958(68)90309-4.
- Shinahr, Ilka (1974), "Two- and three-dimensional firing-squad synchronization problem", *Information and Control* **24** (2): 163–180, doi:10.1016/S0019-9958(74)80055-0.
- Waksman, Abraham (1966), "An optimum solution to the firing squad synchronization problem", *Information and Control* **9** (1): 66–78, doi:10.1016/S0019-9958(66)90110-0.
- Wolfram, Stephen (2002), "Firing squad synchronization" [2], *A New Kind of Science*, Wolfram Media, p. 1035, ISBN 1-57955-008-8.

External links

- Weisstein, Eric W., "Firing Squad Problem [3]" from MathWorld.

References

[1] http://www.cecs.csulb.edu/~daring/research/formulations.pdf

[2] http://www.wolframscience.com/nksonline/page-1035b-text

[3] http://mathworld.wolfram.com/FiringSquadProblem.html

Majority problem (cellular automaton)

The **majority problem**, or **density classification task** is the problem of finding one-dimensional cellular automaton rules that accurately perform majority voting.

Using local transition rules, cells cannot know the total count of all the ones in system. In order to count the number of ones (or, by symmetry, the number of zeros), the system requires a logarithmic number of bits in the total size of the system. It also requires the system send messages over a distance linear in the size of the system and for the system to recognize a non-regular language. Thus, this problem is an important test case in measuring the computational power of cellular automaton systems.

Problem statement

Given a configuration of a two-state cellular automata with $i + j$ cells total, i of which are in the zero state and j of which are in the one state, a correct solution to the voting problem must eventually set all cells to zero if $i > j$ and must eventually set all cells to one if $i < j$. The desired eventual state is unspecified if $i = j$.

The problem can also be generalized to testing whether the proportion of zeros and ones is above or below some threshold other than 50%. In this generalization, one is also given a threshold ρ; a correct solution to the voting problem must eventually set all cells to zero if $\frac{j}{i+j} < \rho$ and must eventually set all cells to one if $\frac{j}{i+j} > \rho$. The desired eventual state is unspecified if $\frac{j}{i+j} = \rho$.

Approximate solutions

Gács, Kurdyumov, and Levin found an automaton that, although it does not always solve the majority problem correctly, does so in many cases.[1] In their approach to the problem, the quality of a cellular automaton rule is measured by the fraction of the 2^{i+j} possible starting configurations that it correctly classifies.

The rule proposed by Gacs, Kurdyumov, and Levin sets the state of each cell as follows. If a cell is 0, its next state is formed as the majority among the values of itself, its immediate neighbor to the left, and its neighbor three spaces to the left. If, on the other hand, a cell is 1, its next state is formed symmetrically, as the majority among the values of itself, its immediate neighbor to the right, and its neighbor three spaces to the right. In randomly generated instances, this achieves about 78% accuracy in correctly determining the majority.

Das, Mitchell, and Crutchfield showed that it is possible to develop better rules using genetic algorithms.[2]

Impossibility of a perfect classifier

In 1994, Land and Belew[3] showed that no two-state rule with radius r and density ρ correctly solves the voting problem on all starting configurations when the number of cells is sufficiently large (larger than about $4r/\rho$).

Their argument shows that because the system is deterministic, every cell surrounded entirely by zeros or ones must then become a zero. Likewise, any perfect rule can never make the ratio of ones go above ρ if it was below (or vice-versa). They then show that any assumed perfect rule will either cause an isolated one that pushed the ratio over ρ to be cancelled out or, if the ratio of ones is less than ρ, will cause an isolated one to introduce spurious ones into a block of zeros causing the ratio of ones to be become greater than ρ.

Exact solution with alternative termination conditions

As observed by Capcarrere, Sipper, and Tomassini,[4] [5] the majority problem may be solved perfectly if one relaxes the definition by which the automaton is said to have recognized the majority. In particular, for the Rule 184 automaton, when run on a finite universe with cyclic boundary conditions, each cell will infinitely often remain in the majority state for two consecutive steps while only finitely many times being in the minority state for two consecutive steps.

Alternatively, a hybrid automaton that runs Rule 184 for a number of steps linear in the size of the array, and then switches to the majority rule (Rule 232), that sets each cell to the majority of itself and its neighbors, solves the majority problem with the standard recognition criterion of either all zeros or all ones in the final state. However, this machine is not itself a cellular automaton.[6]

References

[1] Gács, Péter; Kurdyumov, G. L.; Levin, L. A. (1978). "One dimensional uniform arrays that wash out finite islands" (http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=1551&option_lang=eng) (in Russian). *Problemy Peredachi Informatsii* **14**: 92–98. .

[2] Das, Rajarshi; Crutchfield, J. P.; Mitchell, Melanie; Hanson, J. E. (1995). "Evolving globally synchronized cellular automata" (http://www.cs.pdx.edu/~mm/EGSCA.pdf). In Eshelman, Larry J.. *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco: Morgan Kaufmann. .

[3] Land, Mark; Belew, Richard (1995). "No perfect two-state cellular automata for density classification exists". *Physical Review Letters* **74** (25): 1548–1550. Bibcode 1995PhRvL..74.5148L. doi:10.1103/PhysRevLett.74.5148. PMID 10058695.

[4] Capcarrere, Mathieu S.; Sipper, Moshe; Tomassini, Marco (1996). "Two-state, $r = 1$ cellular automaton that classifies density" (http://www.leitl.org/docs/density.ps.gz). *Phys. Rev. Lett.* **77** (24): 4969–4971. Bibcode 1996PhRvL..77.4969C. doi:10.1103/PhysRevLett.77.4969. PMID 10062680. .

[5] Sukumar, N. (1998). *Effect of boundary conditions on cellular automata that classify density*. arXiv:comp-gas/9804001.

[6] Fukś, Henryk (1997). "Solution of the density classification problem with two cellular automata rules". *Physical Review E* **55** (3): 2081–2084. arXiv:comp-gas/9703001.

Asynchronous cellular automaton

Cellular automata, as with other multi-agent system models, usually treat time as discrete and state updates as occurring synchronously. The state of every cell in the model is updated together, before any of the new states influence other cells. In contrast, an **asynchronous cellular automaton** is able to update individual cells independently, in such a way that the new state of a cell affects the calculation of states in neighbouring cells.

Implementations of synchronous updating can be analysed in two phases. The first, interaction, calculates the new state of each cell based on the neighbourhood and the update rule. State values are held in a temporary store. The second phase updates state values by copying the new states to the cells.

In contrast, asynchronous updating does not necessarily separate these two phases: in the simplest case (fully asynchronous updating), changes in state are implemented immediately. We can summarise this difference as follows

Synchronous phase 1 (interaction):	$\forall i \in N : \tau_i^{(t+1)} = f(\sigma_{k \in K_i}^{(t)})$
Synchronous phase 2 (update):	$\hat{\sigma}^{(t+1)} = \hat{\tau}^{(t+1)}$
Fully Asynchronous updating:	$\forall i \in N : \sigma_i^{(t+1)} = f(\sigma_{k \in K_i}^{(t)})$

where $\hat{\sigma}^{(t)}$ is the vector of states of the elements at time t, $\hat{\tau}^{(t)}$ is a temporary copy used in the updating, i is the index to an individual element, N is the total number of elements in the model, and $f()$ is a function that calculates the new state of an element based on the current state of the elements in set K_i, where $|K_i| \leq N$.
The synchonous approach assumes the presence of a global clock to ensure all cells are updated together. While convenient for preparing computer systems, this is an unrealistic assumption if the model is intended to represent, for example, a living system where there is no evidence of the presence of such a device.

A general method repeatedly discovered independently (by K. Nakamura in the 1970s, by T. Toffoli in the 1980s, and by C. L. Nehaniv in 1998) allows one to emulate exactly the behaviour of a synchronous cellular automaton via an asynchronous one constructed as a simple modification of the synchronous cellular automaton (Nehaniv 2002). Correctness of this method however has only more recently been rigorously proved (Nehaniv, 2004). As a consequence, it follows immediately from results on synchronous cellular automata that asynchronous cellular automata are capable of emulating, e.g., Conway's Game of Life, of universal computation, and of self-replication (e.g., as in a Von Neumann universal constructor). Moreover, the general construction and the proof also applies to the more general class of synchronous automata networks (inhomogeneous networks of automata over directed graphs, allowing external inputs – which includes cellular automata as a special case), showing constructively how their behaviour may be asynchronously realized by a corresponding asynchronous automata network.

Update Schemes

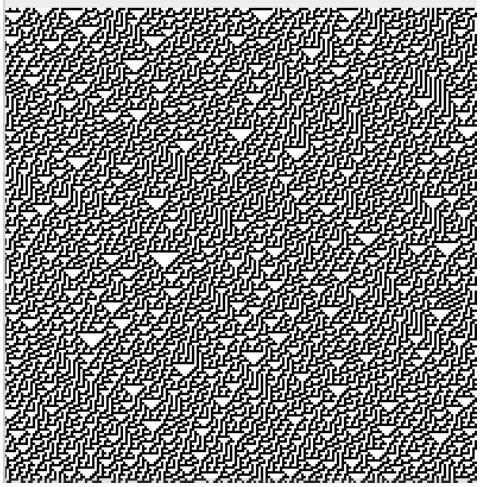
Several studies have implemented asynchronous models and found that their behaviour differs from the synchronous ones. Bersini and Detours (1994) have shown how sensitive Conway's Game of Life is to the updating scheme. Any interesting behaviour disappears in the asynchronous case. Harvey and Bossomaier (1997) pointed out that stochastic updating in random boolean networks results in the expression of point attractors only: there is no repeatable cyclic behaviour, although they introduced the concept of loose cyclic attractors. Kanada (1994) has shown that some one-dimensional CA models that generate non-chaotic patterns when updated synchronously generate edge of chaos patterns when randomised. Orponen (1997) has demonstrated that any synchronously updated network of threshold logic units (see Artificial neuron) can be simulated by a network that has no constraints on the order of updates. Sipper et al. (1997) investigated the evolution of non-uniform CAs that perform specific computing tasks. These models relax the normal requirement of all nodes having the same update rule. In their models, nodes were organised

into blocks. Nodes within a block were updated synchronously, but blocks were updated asynchronously. They experimented with three schemes: (1) at each time step, a block is chosen at random with replacement; (2) at each time step, a block is chosen at random without replacement; (3) at each time step, a block is chosen according to a fixed update order.

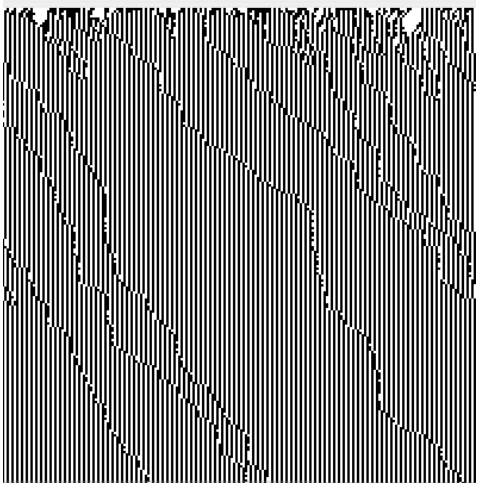
There are different types of asynchronous updating, and different authors have described these in different ways. The schemes shown in the images below are as follows (Cornforth et al. 2005):

- The synchronous scheme - all cells are updated in parallel at each time step. This is the conventional model, stated here for comparison.
- The random independent scheme - at each time step, a cell is chosen at random with replacement, and updated.
- The random order scheme - at each time step, all nodes are updated, but in random order.
- The cyclic scheme - at each time step a node is chosen according to a fixed update order, which was decided at random during initialisation of the model.
- The self-clocked scheme - each cell has an independent timer, initialised to a random period and phase. When the period has expired, the cell is updated and the timer reset. Updating is autonomous and proceeds at different rates for different cells.
- The self-sync scheme - the same as the clocked scheme, but the phase of the timers are affected by local coupling to neighbours, and so are able to achieve local synchrony.

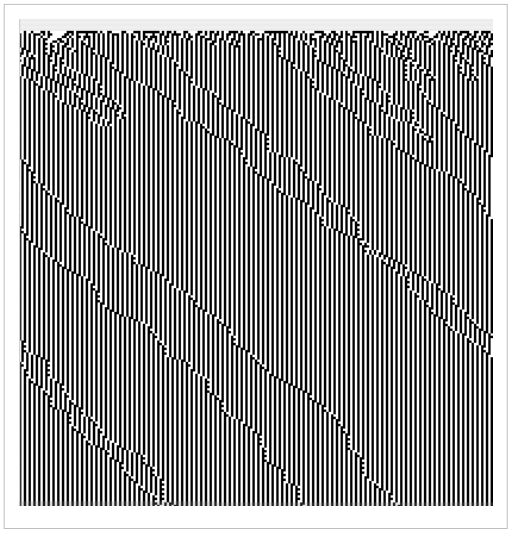
The time-state diagrams below show the differences that are caused by changing the update scheme of the cellular automata model without changing any other parameters. The rule used, rule 30, is the same for each diagram.



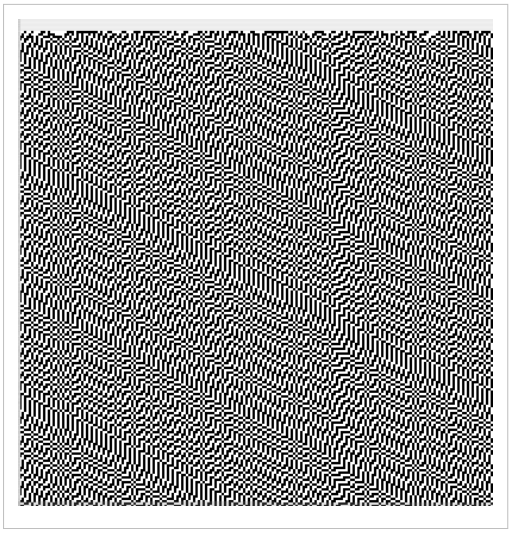
Original rule 30



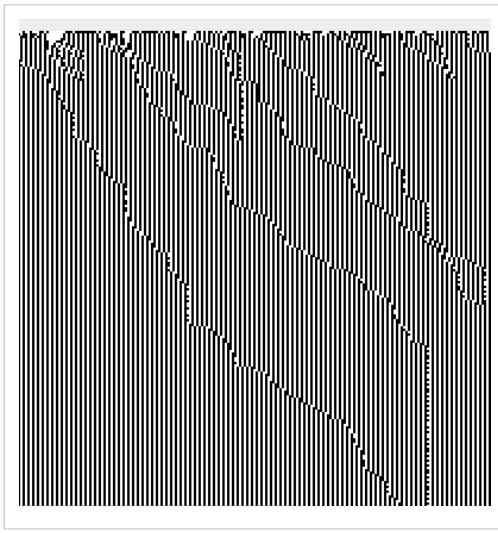
Rule 30 updated randomly



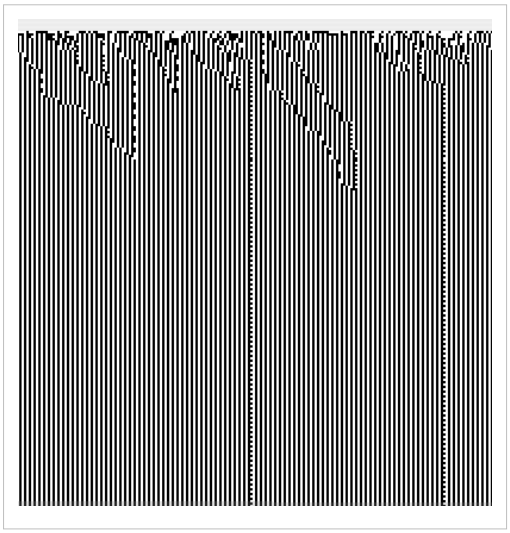
Rule 30 updated in random order



Rule 30 updated in cyclic order



Self-clocked rule 30



Self-synchronous rule 30

Implications

Often, models like cellular automata are used to help understanding of processes that work in real life. By building simplified models, new insights can be learned. There is always a question of how simple these models should be in order to adequately describe what is being modelled. The use of asynchronous models can allow an extra level of realism in the model. All of the schemes described above have their part in real life. The random independent scheme could be appropriate for modelling social networks or communication in computer networks. The clocked scheme could be appropriate for modelling insect colonies, while the self-synchronous scheme could be applied to neural tissue.

References

- H. Bersini and V. Detours, 1994. Asynchrony induces stability in cellular automata based models, *Proceedings of the IVth Conference on Artificial Life* , pages 382-387, Cambridge, MA, July 1994, vol 204, no. 1-2, pp. 70-82.
- Cornforth, D, Green, D, & Newth, D 2005, Ordered Asynchronous Processes in Multi-Agent Systems, *Physica D*, vol 204, no. 1-2, pp. 70-82.
- Cornforth, D, Green, DG, Newth D & Kirley M 2002, Do Artificial Ants March in Step? Ordered Asynchronous Processes and Modularity in Biological Systems [1]. In Standish, Bedau, Abbass, *Proceedings of the Eighth International Conference on Artificial Life*, Sydney, pp. 28-32
- Fatès N., and Morvan M., (2005), An Experimental Study of Robustness to Asynchronism for Elementary Cellular Automata, *Complex Systems*: Volume 16 / Issue 1 , pp. 1-27.
- Fatès N., Morvan M., N. Schabanel, and E. Thierry, (2006), Fully asynchronous behavior of double-quiescent elementary cellular automata, *Theoretical Computer Science*: Volume 362 , pp. 1 - 16.
- Harvey I., and Bossomaier T.R.J, (1997). Time Out of Joint: Attractors in Asynchronous Boolean Networks. In Husbands and Harvey (eds.), *Proceedings of the Fourth European Conference on Artificial Life*, 67-75, MIT Press.
- Kanada Y. (1994). The Effects of Randomness in Asynchronous 1D Cellular Automata [2]. *Artificial Life IV*.
- Nehaniv, C. L. (2002). Evolution in Asynchronous Cellular Automata, *Artificial Life VIII*, 65-73, MIT Press.
- Nehaniv, C. L. (2004). Asynchronous Automata Networks Can Emulate Any Synchronous Automata Network, *International Journal of Algebra & Computation*, 14(5-6):719-739.
- Orponen, P. (1997). Computing with Truly Asynchronous Threshold Logic Networks. *Theoretical Computer Science* 174(1-2):123-136.
- Sipper M, Tomassini M. and Capcarrere M.S. (1997). Evolving Asynchronous and Scalable Non-Uniform Cellular Automata. *Proc. of Intl. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, Springer-Verlag.
- Virtual Laboratory at Monash University [3] Online simulations of asynchronous updating in cellular automata.

References

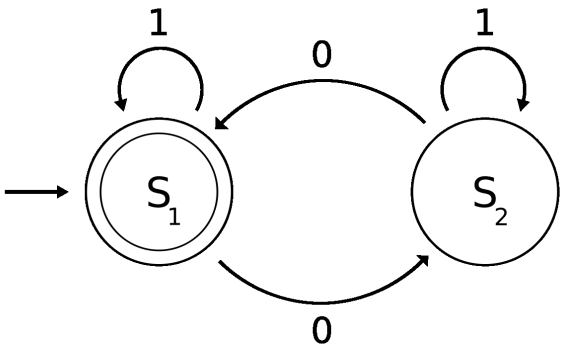
[1] http://web.archive.org/web/20050712082529/www.alife.org/alife8/proceedings/sub1213.pdf

[2] http://www.kanadas.com/CA/AsyncCA/AsyncCA-ALife.Summary.pdf

[3] http://vlab.infotech.monash.edu.au/simulations/cellular-automata/asynchronous-updating

Automata theory

In theoretical computer science, **automata theory** is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata. Automata comes from the Greek word αὐτόματα meaning "self-acting".



An example of automata. The study of the mathematical properties of such automata is automata theory

The figure at right illustrates a finite state machine, which is one well-known variety of automaton. This automaton consists of states (represented in the figure by circles), and transitions (represented by arrows). As the automaton sees a symbol of input, it makes a *transition* (or *jump*) to another state, according to its *transition function* (which takes the current state and the recent symbol as its inputs).

Automata theory is also closely related to formal language theory, as the automata are often classified by the class of formal languages they are able to recognize. An automaton can be a finite representation of a formal language that may be an infinite set.

Automata play a major role in compiler design and parsing.

Automata

Following is an introductory definition of one type of automata, which attempts to help one grasp the essential concepts involved in automata theory.

Informal description

An automaton is supposed to *run* on some given sequence of *inputs* in discrete time steps. At each time step, an automaton gets one input that is picked up from a set of *symbols* or *letters*, which is called an *alphabet*. At any time, the symbols so far fed to the automaton as input form a finite sequence of symbols, which is called a *word*. An automaton contains a finite set of *states*. At each instance in time of some run, the automaton is *in* one of its states. At each time step when the automaton reads a symbol, it *jumps* or *transits* to a next state that is decided by a function that takes current state and the symbol currently read as parameters. This function is called *transition function*. The automaton *reads* the symbols of the input word one after another and *transits* from state to state according to the transition function, until the word is read completely. Once the input word has been read, the automaton is said to have been *stopped* and the state at which automaton has stopped is called *final state*. Depending on the final state, it's said that the automaton either *accepts* or *rejects* an input word. There is a subset of states of the automaton, which is defined as the set of *accepting states*. If the final state is an accepting state, then the automaton *accepts* the word. Otherwise, the word is *rejected*. The set of all the words accepted by an automaton is called the *language recognized by the automaton*.

In short, an automaton is a mathematical object that takes a word as input and decides either to accept it or reject it. Since all computational problems are reducible into the accept/reject question on words (all problem instances can be represented in a finite length of symbols), automata theory plays a crucial role in computational theory.

Formal definition

Automaton

An **automaton** is represented formally by a 5-tuple (Q,Σ,δ,q_0,A), where:

- Q is a finite set of *states*.
- Σ is a finite set of *symbols*, called the *alphabet* of the automaton.
- δ is the **transition function**, that is, $\delta: Q \times \Sigma \to Q$.
- q_0 is the *start state*, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
- A is a set of states of Q (i.e. $A \subseteq Q$) called **accept states**.

Input word

An automaton reads a finite string of symbols $a_1,a_2,....,a_n$, where $a_i \in \Sigma$, which is called an *input word*. The set of all words is denoted by Σ^*.

Run

A *run* of the automaton on an input word $w = a_1,a_2,....,a_n \in \Sigma^*$, is a sequence of states $q_0,q_1,q_2,....,q_n$, where $q_i \in Q$ such that q_0 is the start state and $q_i = \delta(q_{i-1},a_i)$ for $0 < i \leq n$. In words, at first the automaton is at the start state q_0, and then the automaton reads symbols of the input word in sequence. When the automaton reads symbol a_i it jumps to state $q_i = \delta(q_{i-1},a_i)$. q_n is said to be the *final state* of the run.

Accepting word

A word $w \in \Sigma^*$ is accepted by the automaton if $q_n \in A$.

Recognized language

An automaton can recognize a formal language. The language $L \subseteq \Sigma^*$ recognized by an automaton is the set of all the words that are accepted by the automaton.

Recognizable languages

The recognizable languages are the set of languages that are recognized by some automaton. For the above definition of automata the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

Variant definitions of automata

Automata are defined to study useful machines under mathematical formalism. So, the definition of an automaton is open to variations according to the "real world machine", which we want to model using the automaton. People have studied many variations of automata. The most standard variant, which is described above, is called a deterministic finite automaton. The following are some popular variations in the definition of different components of automata.

Input

- *Finite input*: An automaton that accepts only finite sequence of symbols. The above introductory definition only accepts finite words.
- *Infinite input*: An automaton that accepts infinite words (ω-words). Such automata are called *ω-automata*.
- *Tree word input*: The input may be a *tree of symbols* instead of sequence of symbols. In this case after reading each symbol, the automaton *reads* all the successor symbols in the input tree. It is said that the automaton *makes one copy* of itself for each successor and each such copy starts running on one of the successor symbol from the state according to the transition relation of the automaton. Such an automaton is called tree automaton.
- *Infinite tree input* : The two extensions above can be combined, so the automaton reads a tree structure with (in)finite branches. Such an automaton is called infinite tree automaton

States

- *Finite states*: An automaton that contains only a finite number of states. The above introductory definition describes automata with finite numbers of states.
- *Infinite states*: An automaton that may not have a finite number of states, or even a countable number of states. For example, the quantum finite automaton or topological automaton has uncountable infinity of states.
- *Stack memory*: An automaton may also contain some extra memory in the form of a stack in which symbols can be pushed and popped. This kind of automaton is called a *pushdown automaton*

Transition function

- *Deterministic*: For a given current state and an input symbol, if an automaton can only jump to one and only one state then it is a *deterministic automaton*.
- *Nondeterministic*: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation. Notice that the term transition function is replaced by transition relation: The automaton *non-deterministically* decides to jump into one of the allowed choices. Such automata are called *nondeterministic automata*.
- *Alternation*: This idea is quite similar to tree automaton, but orthogonal. The automaton may run its *multiple copies* on the *same* next read symbol. Such automata are called *alternating automata*. Acceptance condition must satisfy all runs of such *copies* to accept the input.

Acceptance condition

- *Acceptance of finite words*: Same as described in the informal definition above.
- *Acceptance of infinite words*: an *omega automaton* cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
- *Probabilistic acceptance*: An automaton need not strictly accept or reject an input. It may accept the input with some probability between zero and one. For example, quantum finite automaton, geometric automaton and *metric automaton* have probabilistic acceptance.

Different combinations of the above variations produce many classes of automaton.

Automata theory

Automata theory is a subject matter which studies properties of various types of automata. For example, the following questions are studied about a given type of automata.

- Which class of formal languages is recognizable by some type of automata? (Recognizable languages)
- Are certain automata *closed* under union, intersection, or complementation of formal languages? (Closure properties)
- How much is a type of automata expressive in terms of recognizing class of formal languages? And, their relative expressive power? (Language Hierarchy)

Automata theory also studies if there exist any effective algorithm or not to solve problems similar to the following list.

- Does an automaton accept any input word? (emptiness checking)
- Is it possible to transform a given non-deterministic automaton into deterministic automaton without changing the recognizable language? (Determinization)
- For a given formal language, what is the smallest automaton that recognizes it? (Minimization).

Classes of automata

The following is an incomplete list of types of automata.

Automata	Recognizable language
Deterministic finite automata (DFA)	regular languages
Nondeterministic finite automata (NFA)	regular languages
Nondeterministic finite automata with ε-transitions (FND-ε or ε-NFA)	regular languages
Pushdown automata (PDA)	context-free languages
Linear bounded automata (LBA)	context-sensitive language
Turing machines	recursively enumerable languages
Timed automata	
Deterministic Büchi automata	ω-limit languages
Nondeterministic Büchi automata	ω-regular languages
Nondeterministic/Deterministic Rabin automata	ω-regular languages
Nondeterministic/Deterministic Streett automata	ω-regular languages
Nondeterministic/Deterministic parity automata	ω-regular languages
Nondeterministic/Deterministic Muller automata	ω-regular languages

Discrete, continuous, and hybrid automata

Normally automata theory describes the states of abstract machines but there are analog automata or continuous automata or hybrid discrete-continuous automata, which use analog data, continuous time, or both.

Applications

Each model in automata theory plays an imporant roles in several applied areas. Finite automata are used in text processing, compilers, and hardware design. Context-free grammar (CFGs) are used in programming languages and artificial intelligence. Originally, CFGs were used in the study of the human languages. Cellular automata are used in the field of biology, the most common example being John Conway's Game of Life. Some other examples which could be explained using automata theory in biology include mollusk and pine cones growth and pigmentation patterns. Going further, a theory suggesting that the whole universe is computed by some sort of a discrete automaton, is advocated by some scientists. The idea originated in the work of Konrad Zuse, most importantly his 1969 book Rechnender Raum, and gives rise to Digital physics.

Connection to Category theory

One can define several distinct categories of automata[1] following the automata classification into different types described in the previous section. The mathematical category of deterministic automata, sequential machines or *sequential automata*, and Turing machines with automata homomorphisms defining the arrows between automata is a Cartesian closed category,[2] [3] it has both categorical limits and colimits. An automata homomorphism maps a quintuple of an automaton A_i onto the quintuple of another automaton A_j.[4] Automata homomorphisms can also be considered as *automata transformations* or as semigroup homomorphisms, when the state space,**S**, of the automaton is defined as a semigroup $\mathbf{S}_g$. Monoids are also considered as a suitable setting for automata in monoidal categories.[5] [6] [7]

Categories of variable automata

One could also define a *variable automaton*, in in the sense of Norbert Wiener in his book on *"Human Use of Human Beings" via* the endomorphisms $A_i \to A_i$. Then, one can show that such variable automata homomorphisms form a mathematical group. In the case of non-deterministic, or other complex kinds of automata, the latter set of endomorphisms may become, however, a *variable automaton groupoid*. Therefore, in the most general case, categories of variable automata of any kind are categories of groupoids[8] or groupoid categories. Moreover, the category of reversible automata is then a 2-category, and also a subcategory of the 2-category of groupoids, or the groupoid category.

References

[1] Jirí Adámek and Vera Trnková. 1990. *Automata and Algebras in Categories*. Kluwer Academic Publishers:Dordrecht and Prague

[2] S. Mac Lane, Categories for the Working Mathematician, Springer, New York (1971)

[3] http://planetmath.org/encyclopedia/CartesianClosedCategory.html Cartesian closed category

[4] http://planetmath.org/encyclopedia/SequentialMachine3.html The Category of Automata

[5] http://www.csee.wvu.edu/~jworthing/asl2010.pdf James Worthington.2010.Determinizing, Forgetting, and Automata in Monoidal Categories. ASL North American Annual Meeting,March 17, 2010

[6] Aguiar, M. and Mahajan, S.2010. *"Monoidal Functors, Species, and Hopf Algebras"*.

[7] Meseguer, J., Montanari, U.: 1990 Petri nets are monoids. *Information and Computation* **88**:105–155

[8] http://en.wikipedia.org/wiki/Groupoid#Category_of_groupoids Category of groupoids

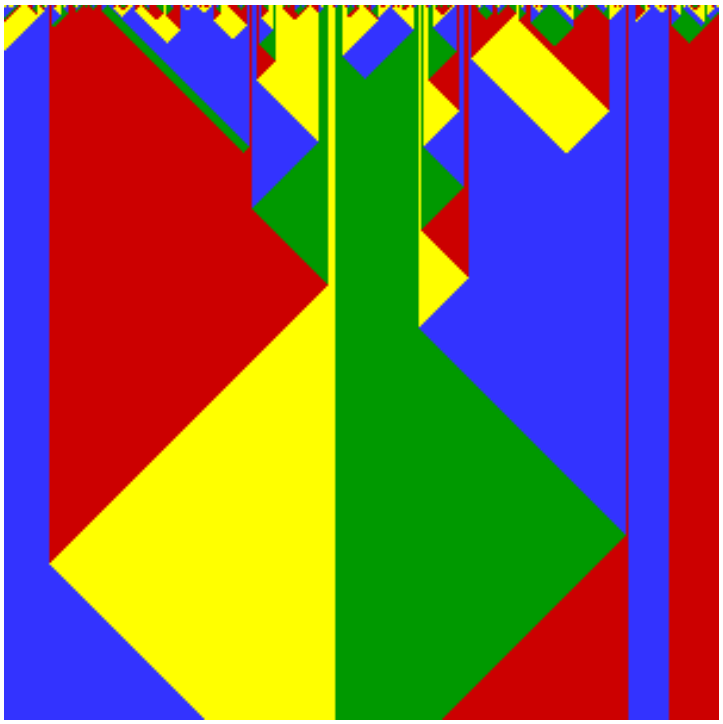
- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Pearson Education. ISBN 0-201-44124-1.
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X. Part One: Automata and Languages, chapters 1–2, pp. 29–122. Section 4.1: Decidable Languages, pp. 152–159. Section 5.1: Undecidable Problems from Language Theory, pp. 172–183.
- James P. Schmeiser, David T. Barnard (1995). *Producing a top-down parse order with bottom-up parsing*. Elsevier North-Holland.

External links

- Visual Automata Simulator (http://www.cs.usfca.edu/~jbovet/vas.html), A tool for simulating, visualizing and transforming finite state automata and Turing Machines, by Jean Bovet
- JFLAP (http://www.jflap.org)
- dk.brics.automaton (http://www.brics.dk/automaton)
- libfa (http://www.augeas.net/libfa/index.html)
- Proyecto SEPa (in Spanish) (http://www.ucse.edu.ar/fma/sepa/)
- Exorciser (in German) (http://www.swisseduc.ch/informatik/exorciser/index.html)
- Automata Made it on Java, with the Sources so you can see (http://torturo.com/programas-hechos-en-java/)

Cyclic cellular automaton

The **cyclic cellular automaton** is a cellular automaton rule developed by David Griffeath and studied by several other cellular automaton researchers. In this system, each cell remains unchanged until some neighboring cell has a modular value exactly one unit larger than that of the cell itself, at which point it copies its neighbor's value. One-dimensional cyclic cellular automata can be interpreted as systems of interacting particles, while cyclic cellular automata in higher dimensions exhibit complex spiraling behavior.



A one-dimensional cyclic cellular automaton with $n = 4$, run for 300 steps from a random initial configuration.

Rules

As with any cellular automaton, the cyclic cellular automaton consists of a regular grid of cells in one or more dimensions. The cells can take on any of n states, ranging from 0 to $n-1$. The first generation starts out with random states in each of the cells. In each subsequent generation, if a cell has a neighboring cell whose value is the successor of the cell's value, the cell is "consumed" and takes on the succeeding value. (Note that 0 is the successor of $n-1$; see also modular arithmetic.) More general forms of this type of rule also include a *threshold* parameter, and only allow a cell to be consumed when the number of neighbors with the successor value exceeds this threshold.

One dimension

The one-dimensional cyclic cellular automaton has been extensively studied by Robert Fisch, a student of Griffeath.[1] Starting from a random configuration with $n = 3$ or $n = 4$, this type of rule can produce a pattern which, when presented as a time-space diagram, shows growing triangles of values competing for larger regions of the grid.

The boundaries between these regions can be viewed as moving particles which collide and interact with each other. In the three-state cyclic cellular automaton, the boundary between regions with values i and $i + 1 \pmod n$ can be viewed as a particle that moves either leftwards or rightwards depending on the ordering of the regions; when a leftward-moving particle collides with a rightward-moving one, they annihilate each other, leaving two fewer particles in the system. This type of ballistic annihilation process occurs in several other cellular automata and related systems, including Rule 184, a cellular automaton used to model traffic flow.[2]

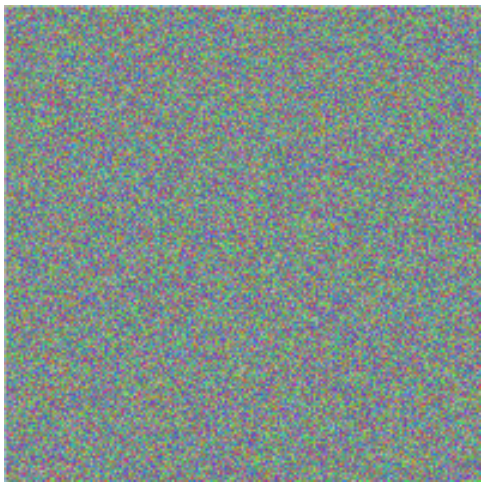
In the $n = 4$ automaton, the same two types of particles and the same annihilation reaction occur. Additionally, a boundary between regions with values i and $i + 2 \pmod n$ can be viewed as a third type of particle, that remains stationary. A collision between a moving and a stationary particle results in a single moving particle moving in the opposite direction.

However, for $n \geq 5$, random initial configurations tend to stabilize quickly rather than forming any non-trivial long-range dynamics. Griffeath has nicknamed this dichotomy between the long-range particle dynamics of the $n = 3$

and $n = 4$ automata on the one hand, and the static behavior of the $n \geq 5$ automata on the other hand, "Bob's dilemma", after Bob Fisch.[3]

Two or more dimensions

In two dimensions, with no threshold and the von Neumann neighborhood or Moore neighborhood, this cellular automaton generates three general types of patterns sequentially, from random initial conditions on sufficiently large grids, regardless of n.[4] At first, the field is purely random. As cells consume their neighbors and get within range to be consumed by higher-ranking cells, the automaton goes to the consuming phase, where there are blocks of color advancing against remaining blocks of randomness. Important in further development are objects called demons, which are cycles of adjacent cells containing one cell of each state, in the cyclic order; these cycles continuously rotate and generate waves that spread out in a spiral pattern centered at the cells of the demon. The third stage, the demon stage, is dominated by these cycles. Almost surely, every cell of the automaton eventually enters a repeating cycle of states, where the period of the repetition is either n or (for automata with n odd and the von Neumann neighborhood) $n + 1$. The same eventually-period behavior occurs also in higher dimensions. Small structures can also be constructed with any even period between n and $3n/2$. Merging these structures, configurations can be built with a global super-polynomial period.[5]



Animation of two dimensional cyclical cellular automaton growing to repeating patterns from a random beginning.

A two-dimensional cyclic cellular automaton with $n = 16$, after 400 steps starting from a random initial configuration. All three types of patterns formed by this automaton are visible in this image.

For larger neighborhoods, similar spiraling behavior occurs for low thresholds, but for sufficiently high thresholds the automaton stabilizes in the block of color stage without forming spirals. At intermediate values of the threshold, a complex mix of color blocks and partial spirals, called turbulence, can form.[6] For appropriate choices of the number of states and the size of the neighborhood, the spiral patterns formed by this automaton can be made to resemble those of the

Belousov-Zhabotinsky reaction in chemistry, although other cellular automata more accurately model the excitable medium that leads to this reaction.

Notes

[1] Fisch (1990a,1990b,1992).
[2] Belitsky and Ferrari (2005).
[3] Bob's Dilemma (http://psoup.math.wisc.edu/archive/recipe29.html). Recipe 29 in David Griffeath's Primordial Soup Kitchen.
[4] Bunimovich and Troubetzkoy (1994); Dewdney (1989); Fisch, Gravner, and Griffeath (1992); Shalizi and Shalizi (2003); Steif (1995).
[5] Matamala and Moreno (2004)
[6] Turbulent Equilibrium in a Cyclic Cellular Automaton (http://psoup.math.wisc.edu/archive/recipe6.html). Recipe 6 in David Griffeath's Primordial Soup Kitchen.

References

- Belitzky, Vladimir; Ferrari, Pablo A. (1995). "Ballistic annihilation and deterministic surface growth". *Journal of Statistical Physics* **80** (3–4): 517–543. doi:10.1007/BF02178546.
- Bunimovich L. A.; Troubetzkoy, S. E. (1994). "Rotators, periodicity, and absence of diffusion in cyclic cellular automata". *Journal of Statistical Physics* **74** (1–2): 1–10. doi:10.1007/BF02186804.
- Dewdney, A. K. (1989). "Computer Recreations: A cellular universe of debris, droplets, defects, and demons". *Scientific American* (August): 102–105.
- Fisch, R. (1990a). "The one-dimensional cyclic cellular automaton: A system with deterministic dynamics that emulates an interacting particle system with stochastic dynamics". *Journal of Theoretical Probability* **3** (2): 311–338. doi:10.1007/BF01045164.
- Fisch, R. (1990b). "Cyclic cellular automata and related processes". *Physica D* **45** (1–3): 19–25. doi:10.1016/0167-2789(90)90170-T. Reprinted in Gutowitz, Howard A. (ed.) (1991). *Cellular Automata: Theory and Experiment*. MIT Press/North-Holland. pp. 19–25. ISBN 0-262-57086-6.
- Fisch, R. (1992). "Clustering in the one-dimensional three-color cyclic cellular automaton". *Annals of Probability* **20** (3): 1528–1548. doi:10.1214/aop/1176989705.
- Fisch, R.; Gravner, J.; Griffeath, D. (1991). "Threshold-Range Scaling of Excitable Cellular Automata". *Statistics and Computing* **1**: 23–39. doi:10.1007/BF01890834.
- Matamala, Martín; Moreno, Eduardo (2004). "Dynamic of cyclic automata over Z^2". *Theoretical Computer Science* **322** (2): 369–381. doi:10.1016/j.tcs.2004.03.018.
- Shalizi, Cosma Rohilla; Shalizi, Kristina Lisa (2003). "Quantifying self-organization in cyclic cellular automata". In Lutz Schimansky-Geier, Derek Abbott, Alexander Neiman and Christian Van den Broeck (eds.). *Noise in Complex Systems and Stochastic Dynamics*. Bellingham, Washington: SPIE. pp. 108–117. arXiv:nlin/0507067.
- Steif, Jeffrey E. (1995). "Two applications of percolation to cellular automata". *Journal of Statistical Physics* **78** (5–6): 1325–1335. doi:10.1007/BF02180134.

Excitable medium

An **excitable medium** is a nonlinear dynamical system which has the capacity to propagate a wave of some description, and which cannot support the passing of another wave until a certain amount of time has passed (known as the refractory time).

A forest is an example of an excitable medium: if a wildfire burns through the forest, no fire can return to a burnt spot until the vegetation has gone through its refractory period and regrown. In Chemistry, oscillating reactions are excitable media, for example the Belousov-Zhabotinsky reaction and the Briggs-Rauscher reaction. Pathological activities in the heart and brain can be modelled as excitable media. A group of spectators at a sporting event are an excitable medium, as can be observed in a Mexican wave (so-called from its initial appearance in the 1986 World Cup in Mexico).

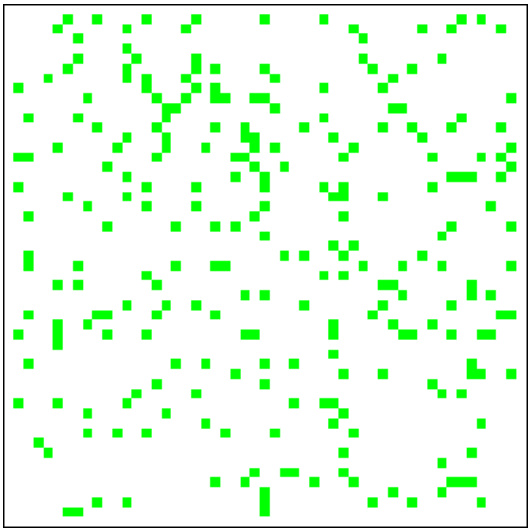
Modelling excitable media

Excitable media can be modelled using both partial differential equations and cellular automata.

With cellular automata

Cellular automata provide a simple model to aid in the understanding of excitable media. Each cell of the automaton is made to represent some section of the medium being modelled (for example, a patch of trees in a forest, or a segment of heart tissue). Each cell can be in one of the three following states:

- Quiescent or excitable — the cell is unexcited, but can be excited. In the forest fire example, this corresponds to the trees being unburnt.
- Excited — the cell is excited. The trees are on fire.
- Refractory — the cell has recently been excited and is temporarily not excitable. This corresponds to a patch of land where the trees have burnt and the vegetation has yet to regrow.



Traveling waves in a model of an excitable medium (White - Quiescent, Green - Excited, Yellow - Refractory)

As in all cellular automata, the state of a particular cell in the next time step depends on the state of the cells around it—its neighbours—at the current time. In an excitable medium the general evolution function that governs the transitions between the three states described above is as follows:

- If a cell is quiescent, then it remains quiescent unless one or more of its neighbours is excited. In the forest fire example, this means a patch of land only burns if a neighbouring patch is on fire.
- If a cell is excited, it becomes refractory at the next iteration. After trees have finished burning, the patch of land is left barren.
- If a cell is refractory, then its remaining refractory period is lessened at the next period, until it reaches the end of the refractory period and becomes excitable once more. The trees regrow.

This function can be refined according to the particular medium. For example, the effect of wind can be added to the model of the forest fire.

Geometries of waves

One-dimensional waves

It is most common for a one-dimensional medium to form a closed circuit, i.e. a ring. For example, the Mexican wave can be modeled as a ring going around the stadium. If the wave moves in one direction it will eventually return to where it started. If, upon a wave's return to the origin, the original spot has gone through its refractory period, then the wave will propagate along the ring again (and will do so indefinitely). If, however, the origin is still refractory upon the wave's return, the wave will be stopped.

In the Mexican wave, for example, if for some reason, the originators of the wave are still standing upon its return it will not continue. If the originators have sat back down then the wave can, in theory, continue.

Two-dimensional waves

Several forms of waves can be observed in a two-dimensional medium.

A *spreading wave* will originate at a single point in the medium and spread outwards. For example a forest fire could start from a lightning strike at the centre of a forest and spread outwards.

A *spiral wave* will again originate at a single point, but will spread in a spiral circuit. Spiral waves are believed to underlie phenomena such as tachycardia and fibrillation.

Spiral waves constitute one of the mechanisms of fibrillation when they organize in long-lasting reentrant activities named rotors.

References

- Leon Glass and Daniel Kaplan, *Understanding Nonlinear Dynamics*.

External links

- An introduction to excitable media [1]
- Java applets that show excitable media in 0, 1, and 2D [2]
- Another excitable medium java applet [3].
- CAPOW software by Rudy Rucker contains several excitable media models [4].

References

[1] http://www.cnd.mcgill.ca/bios/bub/excitablemain.html
[2] http://thevirtualheart.org/appletsindex.html
[3] http://www.joakimlinde.se/java/excitablemedia/index.php
[4] http://www.cs.sjsu.edu/faculty/rucker/capow/

A New Kind of Science

A New Kind of Science	
Author(s)	Stephen Wolfram
Country	US
Language	English
Publisher	Wolfram Media
Publication date	2002
Media type	Print
Pages	1197
ISBN	ISBN 1-57955-008-8

A New Kind of Science is a book by Stephen Wolfram, published in 2002. It contains an empirical and systematic study of computational systems such as cellular automata. Wolfram calls these systems *simple programs* and argues that the scientific philosophy and methods appropriate for the study of simple programs are relevant to other fields of science.

Contents

Computation and its implications

The thesis of *A New Kind of Science* is twofold: that the nature of computation must be explored experimentally, and that the results of these experiments have great relevance to understanding the natural world, which is assumed to be digital. Since its crystallization in the 1930s, computation has been primarily approached from two traditions: engineering, which seeks to build practical systems using computations; and mathematics, which seeks to prove theorems about computation (albeit already in the 1970s computing as a discipline was described as being at the intersection of mathematical, engineering, and empirical/scientific traditions[1] [2]).

Wolfram describes himself as introducing a third major tradition, which is the systematic, empirical investigation of computational systems for their own sake. This is where the "New" and "Science" parts of the book's title originate. However, in proceeding with a scientific investigation of computational systems, Wolfram eventually came to the conclusion that an entirely new method is needed. In his view, traditional mathematics was failing to describe the complexity seen in these systems meaningfully. Through a combination of experiment and theoretical positioning, the book introduces a method that Wolfram argues is the most realistic way to make scientific progress with computational systems, casting *A New Kind of Science* as a "kind" of science, and allows its principles to be potentially applicable in a wide range of fields.

Simple programs

The basic subject of Wolfram's "new kind of science" is the study of simple abstract rules—essentially, elementary computer programs. In almost any class of computational system, one very quickly finds instances of great complexity among its simplest cases. This seems to be true regardless of the components of the system and the details of its setup. Systems explored in the book include cellular automata in one, two, and three dimensions; mobile automata; Turing machines in 1 and 2 dimensions; several varieties of substitution and network systems; primitive recursive functions; nested recursive functions; combinators; tag systems; register machines; reversal-addition; and a number of other systems. For a program to qualify as simple, there are several benchmarks:

1. Its operation can be completely explained by a simple graphical illustration.

2. It can be completely explained in a few sentences of human language.
3. It can be implemented in a computer language using just a few lines of code.
4. The number of its possible variations is small enough so that all of them can be computed.

Generally, simple programs tend to have a very simple abstract framework. Simple cellular automata, Turing machines, and combinators are examples of such frameworks, while more complex cellular automata do not necessarily qualify as simple programs. It is also possible to invent new frameworks, particularly to capture the operation of natural systems. The remarkable feature of simple programs is that a significant percentage of them are capable of producing great complexity. Simply enumerating all possible variations of almost any class of programs quickly leads one to examples that do unexpected and interesting things. This leads to the question: if the program is so simple, where does the complexity come from? In a sense, there is not enough room in the program's definition to directly encode all the things the program can do. Therefore, simple programs can be seen as a minimal example of emergence. A logical deduction from this phenomenon is that if the details of the program's rules have little direct relationship to its behavior, then it is very difficult to directly engineer a simple program to perform a specific behavior. An alternative approach is to try to engineer a simple overall computational framework, and then do a brute-force search through all of the possible components for the best match.

Simple programs are capable of a remarkable range of behavior. Some have been proven to be universal computers. Others exhibit properties familiar from traditional science, such as thermodynamic behavior, continuum behavior, conserved quantities, percolation, sensitive dependence on initial conditions, and others. They have been used as models of traffic, material fracture, crystal growth, biological growth, and various sociological, geological, and ecological phenomena. Another feature of simple programs is that making them more complicated seems to have little effect on their overall complexity. *A New Kind of Science* argues that this is evidence that simple programs are enough to capture the essence of almost any complex system.

Mapping and mining the computational universe

In order to study simple rules and their often complex behaviour, Wolfram believes it is necessary to systematically explore all of these computational systems and document what they do. He believes this study should become a new branch of science, like physics or chemistry. The basic goal of this field is to understand and characterize the computational universe using experimental methods.

The proposed new branch of scientific exploration admits many different forms of scientific production. For instance, qualitative classifications like those found in biology are often the results of initial forays into the computational jungle. On the other hand, explicit proofs that certain systems compute this or that function are also admissible. There are also some forms of production that are in some ways unique to this field of study. For instance, the discovery of computational mechanisms that emerge in different systems but in bizarrely different forms.

Another kind of production involves the creation of programs for the analysis of computational systems—for in the NKS framework, these themselves should be simple programs, and subject to the same goals and methodology. An extension of this idea is that the human mind is itself a computational system, and hence providing it with raw data in as effective a way as possible is crucial to research. Wolfram believes that programs and their analysis should be visualized as directly as possible, and exhaustively examined by the thousands or more. Since this new field concerns abstract rules, it can in principle address issues relevant to other fields of science. However, in general Wolfram's idea is that novel ideas and mechanisms can be discovered in the computational universe—where they can be witnessed in their clearest forms—and then other fields can pick and choose among these discoveries for those they find relevant.

Systematic abstract science

While Wolfram promotes simple programs as a scientific discipline, he also insists that its methodology will revolutionize essentially every field of science. The basis for his claim is that the study of simple programs is the most minimal possible form of science, which is equally grounded in both abstraction and empirical experimentation. Every aspect of the methodology advocated in NKS is optimized to make experimentation as direct, easy, and meaningful as possible—while maximizing the chances that the experiment will do something unexpected. Just as NKS allows computational mechanisms to be studied in their cleanest forms, Wolfram believes the process of doing NKS captures the essence of the process of doing science—and allows that process's strengths and shortcomings to be directly revealed.

Wolfram believes that the computational realities of the universe make science hard for fundamental reasons. But he also argues that by understanding the importance of these realities, we can learn to leverage them in our favor. For instance, instead of reverse engineering our theories from observation, we can simply enumerate systems and then try to match them to the behaviors we observe. A major theme of NKS style research is investigating the structure of the possibility space. Wolfram feels that science is far too ad hoc, in part because the models used are too complicated and/or unnecessarily organized around the limited primitives of traditional mathematics. Wolfram advocates using models whose variations are enumerable and whose consequences are straightforward to compute and analyze.

Philosophical underpinnings

Wolfram believes that one of his achievements is not just exclaiming, "computation is important!", but in providing a coherent system of ideas that justifies computation as an organizing principle of science. For instance, Wolfram's concept of *computational irreducibility*—that some complex computations cannot be short-cutted or "reduced", is ultimately the reason why computational models of nature must be considered, in addition to traditional mathematical models. Likewise, his idea of intrinsic randomness generation—that natural systems can generate their own randomness, rather than using chaos theory or stochastic perturbations—implies that explicit computational models may in some cases provide more accurate and rich models of random-looking systems.

Based on his experimental results, Wolfram has developed the *Principle of Computational Equivalence* (see below), which asserts that almost all processes that are not obviously simple are of equivalent sophistication. From this seemingly vague single principle Wolfram draws a broad array of concrete deductions that reinforce many aspects of his theory. Possibly the most important among these is an explanation as to why we experience randomness and complexity: often, the systems we analyze are just as sophisticated as we are. Thus, complexity is not a special quality of systems, like for instance the concept of "heat", but simply a label for all systems whose computations are sophisticated. Understanding this makes the "normal science" of the NKS paradigm possible.

At the deepest level, Wolfram believes that like many of the most important scientific ideas, the Principle allows science to be more general by pointing out new ways in which humans are not special. In recent times, it has been thought that the complexity of human intelligence makes us special—but the Principle asserts otherwise. In a sense, many of Wolfram's ideas are based on understanding the scientific process—including the human mind—as operating within the same universe it studies, rather than somehow being outside it.

Principle of computational equivalence

The principle states that systems found in the natural world can perform computations up to a maximal ("universal") level of computational power. Most systems can attain this level. Systems, in principle, compute the same things as a computer. Computation is therefore simply a question of translating input and outputs from one system to another. Consequently, most systems are computationally equivalent. Proposed examples of such systems are the workings of the human brain and the evolution of weather systems.

Applications and results

There are a vast number of specific results and ideas in the NKS book, and they can be organized into several themes. One common theme of examples and applications is demonstrating how little it takes to achieve interesting behavior, and how the proper methodology can discover these cases.

First, there are perhaps several dozen cases where the NKS book introduces the simplest known system in some class that has a particular characteristic. Some examples include the first primitive recursive function that results in complexity, the smallest universal Turing Machine, and the shortest axiom for propositional calculus. In a similar vein, Wolfram also demonstrates a large number of minimal examples of how simple programs exhibit phenomena like phase transitions, conserved quantities and continuum behavior and thermodynamics that are familiar from traditional science. Simple computational models of natural systems like shell growth, fluid turbulence, and phyllotaxis are a final category of applications that fall in this theme.

Another common theme is taking facts about the computational universe as a whole and using them to reason about fields in a holistic way. For instance, Wolfram discusses how facts about the computational universe inform evolutionary theory, SETI, free will, computational complexity theory, and philosophical fields like ontology, epistemology, and even postmodernism.

Wolfram suggests that the theory of computational irreducibility may provide a resolution to the existence of free will in a nominally deterministic universe. He posits that the computational process in the brain of the being with free will is actually complex enough so that it cannot be captured in a simpler computation, due to the principle of computational irreducibility. Thus while the process is indeed deterministic, there is no better way to determine the being's will than to essentially run the experiment and let the being exercise it.

The book also contains a vast number of individual results—both experimental and analytic—about what a particular automaton computes, or what its characteristics are, using some methods of analysis.

One specific new technical result in the book is a description of the Turing completeness of the Rule 110 cellular automaton. Rule 110 can be simulated by very small Turing machines, and such a 2-state 5-symbol universal Turing machine is given. Wolfram also conjectures that a particular 2-state 3-symbol Turing machine is universal. In 2007, as part of commemorating the fifth anniversary of the book, a $25,000 prize was offered for a proof of the (2, 3) machine's universality.[3]

NKS Summer School

Every year, Wolfram and his group of instructors[4] organizes a summer school.[5] The first four summer schools from 2003 to 2006 were held at Brown university. Later the summer school was hosted by the university of Vermont at Burlington with the exception of the year 2009 that was held at the Istituto di Scienza e Tecnologie dell'Informazione of the CNR in Pisa, Italy. After seven consecutive summer schools more than 200 people have participated, some of which continued developing their 3-week research projects as their Master's or Ph.D thesis.[6] Some of the research done in the summer school has yielded important published results.[7] [8]

Reception

A New Kind of Science received extensive media publicity for a scientific book, generating scores of articles in such publications as *The New York Times*,[9] *Newsweek*,[10] *Wired*,[11] and *The Economist*.[12] It was a best-seller and won numerous awards. *NKS* was reviewed in a large range of scientific journals, and several themes emerged in these reviews. Many reviewers enjoyed the quality of the book's production and the clear way Wolfram presented many ideas.[13] Even those reviewers who engaged in other criticisms found aspects of the book to be interesting and thought-provoking. On the other hand, many reviewers criticized Wolfram for his lack of modesty, poor editing, lack of mathematical rigor, and the lack of immediate utility of his ideas. Concerning the ultimate importance of the book, a common attitude was that of either skepticism or "wait and see". Many reviewers and the media focused on the use

of simple programs (cellular automata in particular) to model nature, rather than the more fundamental idea of systematically exploring the universe of simple programs.

Scientific philosophy

A key tenet of NKS is that the simpler the system, the more likely a version of it will recur in a wide variety of more complicated contexts. Therefore, NKS argues that systematically exploring the space of simple programs will lead to a base of reusable knowledge. However, many scientists believe that of all possible parameters, only some actually occur in the universe; that, for instance, of all possible variations of an equation, most will be essentially meaningless. NKS has also been criticized for asserting that the behavior of simple systems is somehow representative of all systems.

Methodology

A common criticism of NKS is that it does not follow established scientific methodology. NKS does not establish rigorous mathematical definitions,[14] nor does it attempt to prove theorems.[15] Along these lines, NKS has also been criticized for being heavily visual, with much information conveyed by pictures that do not have formal meaning. It has also been criticized for not using modern research in the field of complexity, particularly the works that have studied complexity from a rigorous mathematical perspective.

Critics also note that none of the book's contents were published in peer-reviewed journals, the standard method for distributing new results, and complain it insufficiently credited other scientists whose work it is built on. Wolfram relegates all discussion of other people to his lengthy endnotes and thus no one is directly credited in the text. His critics argue that even the endnotes are misleading, glossing over many relevant discoveries and thus making Wolfram's work seem more novel.

Utility

NKS has been criticized for not providing specific results that would be immediately applicable to ongoing scientific research. There has also been criticism, implicit and explicit, that the study of simple programs has little connection to the physical universe, and hence is of limited value. Steven Weinberg has pointed out that no real world system has been explained using Wolfram's methods in a satisfactory fashion.[16]

Principle of computational equivalence

The PCE has been criticized for being vague, unmathematical, and for not making directly verifiable predictions; however, Wolfram's group has described the principle as such, not a law, theorem or formula. It has also been criticized for being contrary to the spirit of research in mathematical logic and computational complexity theory, which seek to make fine-grained distinctions between levels of computational sophistication. Others suggest it is little more than a rechristening of the Church-Turing thesis. However, the Church-Turing thesis imposes an upper limit while Wolfram's PCE suggests the nonexistence of intermediate degrees of computation sending a computational system either to the upper level (universal) or to the lowest degree, explained by Klaus Sutner[17] in terms of physics-like computation as a zero-one law claiming that, in practice, constructing actual computers with intermediate degrees is highly artificial and hasn't ever been done, hence endorsing Wolfram's intuition captured in his PCE.

The fundamental theory (NKS Chapter 9)

Wolfram's speculations of a direction towards a fundamental theory of physics have been criticized as vague and obsolete. Scott Aaronson, Assistant Professor of Electrical Engineering and Computer Science at MIT, also claims that Wolfram's methods cannot be compatible with both special relativity and Bell's theorem violations, which conflicts with the observed results of Bell test experiments.[18]

In a 2002 review of NKS, the Nobel laureate and elementary particle physicist Steven Weinberg wrote, "Wolfram himself is a lapsed elementary particle physicist, and I suppose he can't resist trying to apply his experience with digital computer programs to the laws of nature. This has led him to the view (also considered in a 1981 paper by Richard Feynman) that nature is discrete rather than continuous. He suggests that space consists of a set of isolated points, like cells in a cellular automaton, and that even time flows in discrete steps. Following an idea of Edward Fredkin, he concludes that the universe itself would then be an automaton, like a giant computer. It's possible, but I can't see any motivation for these speculations, except that this is the sort of system that Wolfram and others have become used to in their work on computers. So might a carpenter, looking at the moon, suppose that it is made of wood."[19]

According to NKS Chapter 9, special relativity theory and quantum field theory are merely approximations to a digital network with inaccessible signal propagation below the Planck scale. NKS Chapter 9 and M-theory both attempt to unify general relativity theory and quantum field theory. M-theory postulates that there is a minimum physical wavelength and that vibrating string-like entities can model all of physics. NKS Chapter 9 postulates that there is a finite automaton that builds time, space, and energy from informational substrate below the Planck scale. According to Wolfram, infinities and infinitesimals do not occur in nature, except perhaps for time as a potential infinity. In particular, there is a maximum physical wavelength in addition to the minimum physical wavelength postulated by M-theory.

In the NKS theory, the basic physical realities of time, space, and energy are merely approximations that arise from a few simple rules that operate with hidden determinism below the Planck scale. According to Wolfram, "building on the discovery that even simple programs can yield highly complex behavior, *A New Kind of Science* shows that with appropriate kinds of rules, simple programs can give rise to behavior that reproduces a remarkable range of known features of our universe — leading to the bold assertion that there could be a simple short program that represents a truly fundamental model of the universe, and which if run for long enough would reproduce the behavior of our world in every detail."[20]

Natural selection

Wolfram's claim that natural selection is not the fundamental cause of complexity in biology has led some to state that Wolfram does not understand the theory of evolution.[21] However, some experts have acknowledged that natural selection leaves many unanswered questions, which information theory might be able to explain.[22] [23] In this context, Wolfram's work is similar to that of D'Arcy Thompson. D'Arcy Thompson's work, however, is mathematical in nature, while Wolfram's is rule-based (computational). Whereas D'Arcy Thompson showed that nature made certain mathematical choices, without having specified the actual process involved, Wolfram's work suggests that Nature makes these mathematical choices because it is mining what he calls the computational universe from where it picks a computer program.

Originality and self-image

NKS has been heavily criticized as not being original or important enough to justify its title and claims, mostly by people who argue that the book is about simple systems generating complex behavior. However, even though the fact that simple systems are capable of complicated behavior is an important part of the book, the main contribution is the new methodology of mining the computational universe. Edward Fredkin and Konrad Zuse pioneered the idea of a computable universe, the former by writing a line in his book on how the world might be like a cellular

automaton, and later further developed by Fredkin using a toy model called Salt.[24] It has been claimed that NKS tries to take these ideas as its own. This has been mainly suggested by people thinking that Wolfram's main thesis is that the universe is a cellular automaton in spite of the fact that Wolfram's proposal as a discrete model of the universe is a trivalent network. Wolfram himself considers that a cellular automaton model is unsuitable to describe quantum and relativistic properties of nature, as explained in his NKS book.

Jürgen Schmidhuber has also charged that his work on Turing machine-computable physics was stolen without attribution, namely his idea on enumerating possible Turing-computable universes.

Additionally, the core idea that very simple rules often generate great complexity is already an established idea in science, particularly in chaos theory and complex systems research - and to some researchers, this field is considered well-understood. The authoritative manner in which NKS presents a vast number of examples and arguments has been criticized as leading the reader to believe that each of these ideas was original to Wolfram, however the notes section at the end of his book acknowledges many of the discoveries made by these other scientists citing their names together with historical facts, although not in the form of a traditional bibliography section. This is generally considered insufficient in scientific literature, however - end notes are normally reserved for only indirectly related material, and lay readers typically ignore end notes, resulting in the impression that the author performed all the work.

In particular, one of the most substantial new technical results presented in the book, that the rule 110 cellular automaton is Turing complete, was not proven by Wolfram, but by his research assistant, Matthew Cook. This is not particularly a surprise since as explained by Wolfram himself, the book was actually a kind of project involving a group of his research assistants led by Wolfram himself, something that means he didn't do every single experiment or contribution directly, as he also acknowledges in the book. The research assistants were however paid for this work as hired by Wolfram's company, not by a university.

Some have argued that the use of computer simulation is ubiquitous, and instead of starting a paradigm shift NKS just adds justification to a paradigm shift that has been undertaken. Wolfram's NKS might then seem as the book explicitly describing this shift.

References

[1] Wegner, Peter (1976). "Research Paradigms in Computer Science". *Proceedings of the 2nd International Conference on Software Engineering*. San Francisco, CA, USA: IEEE Press. pp. 322–330.

[2] Denning, Peter J.; et al. (1989). "Computing as a Discipline". *Communications of the ACM* **32** (1): 9–23. doi:10.1145/63238.63239.

[3] "The Wolfram 2,3 Turing Machine Research Prize" (http://www.wolframscience.com/prizes/tm23/). . Retrieved 2011-03-31.

[4] http://www.wolframscience.com/summerschool/2009/faculty.html

[5] http://www.wolframscience.com/summerschool/

[6] http://www.wolframscience.com/summerschool/2006/participants/letourneau.html

[7] Rowland (2008). "A natural prime-generating recurrence". *Journal of Integer Sequences .2.8* **11** (08). arXiv:0710.3217.

[8] http://www.springerlink.com/content/m624350kj28305u9/

[9] Johnson, George (9 June 2002). "'A New Kind of Science': You Know That Space-Time Thing? Never Mind" (http://www.nytimes.com/2002/06/09/books/review/09JOHNSOT.html). The New York Times. . Retrieved 28 May 2009.

[10] Levy, Stephen (27 May 2002). "Great Minds, Great Ideas" (http://www.newsweek.com/id/64625). Newsweek. . Retrieved 28 May 2009.

[11] Levy, Stephen (June 2002). "The Man Who Cracked The Code to Everything ..." (http://www.wired.com/wired/archive/10.06/wolfram.html). Wired magazine. . Retrieved 28 May 2009.

[12] "The science of everything" (http://www.economist.com/printedition/displayStory.cfm?Story_ID=1154164). The Economist. 30 May 2002. . Retrieved 28 May 2009.

[13] Rucker, Rudy (November 2003). "Review: A New Kind of Science" (http://sjsu.rudyrucker.com/~rudy.rucker/wolfram_review_AMM_11_2003.pdf). *American Mathematical Monthly*: 851–861. . Retrieved 28 May 2009.

[14] Bailey, David (September 2002). "A Reclusive Kind of Science" (http://crd.lbl.gov/~dhbailey/dhbpapers/dhb-wolfram.pdf). *Computing in Science and Engineering*: 79–81. . Retrieved 28 May 2009.

[15] http://www.ams.org/notices/200302/fea-gray.pdf

[16] Weiss, Peter (2003). "In search of a scientific revolution: controversial genius Stephen Wolfram presses onward" (http://findarticles.com/p/articles/mi_m1200/is_7_164/ai_107699603/?tag=content;col1). *Science News*. .

[17] http://portal.acm.org/citation.cfm?id=771538

[18] http://www.scottaaronson.com/papers/nks.ps
[19] Weinberg, S. (24 October 2002). "Is the Universe a Computer?" (http://www.nybooks.com/articles/15762). *The New York Review of Books*. .
[20] http://wolframscience.com/reference/quick_takes.html
[21] Lavers, Chris (3 August 2002). "How the cheetah got his spots" (http://www.guardian.co.uk/Archive/Article/0,4273,4473834,00.html). London: The Guardian. . Retrieved 28 May 2009.
[22] Wicken, Jeffrey S. (1987). Evolution, Thermodynamics, and Information: Extending the Darwinian Program. Oxford University Press.
[23] Does information theory explain biological evolution? G. Battail 1997 Europhys. Lett. 40
[24] http://www.math.usf.edu/~eclark/ANKOS_zuse_fredkin_thesis.html

External links

- Wolfram, Stephen, *A New Kind of Science* (http://www.wolframscience.com/nksonline). Wolfram Media, Inc., May 14, 2002. ISBN 1-57955-008-8
- Wolfram Science (http://www.wolframscience.com/) the official website, including free online access to full text
- WolframTones: An Experiment in a New Kind of Music (http://tones.wolfram.com/)
- The NKS Blog (http://thenksblog.wordpress.com/)
- InformationSpace (http://www.softcentral.com/informationspace/). Causal set exploration tool which supports 1 dimensional causal sets such as those found in the book.

Quantum cellular automata

Quantum Cellular Automata (QCA) refers to models of quantum computation, which have been devised in analogy to conventional models of cellular automata introduced by von Neumann. It may also refer to quantum dot cellular automata, which is a proposed physical implementation of "classical" cellular automata by exploiting quantum mechanical phenomena.

Usage of the term

In the context of models of computation or of physical systems, *quantum cellular automaton* refers to the merger of elements of both (1) the study of cellular automata in conventional computer science and (2) the study of quantum information processing. In particular, the following are features of models of quantum cellular automata:

- The computation is considered to come about by parallel operation of multiple computing devices, or **cells**. The cells are usually taken to be identical, finite-dimensional quantum systems (e.g. each cell is a qubit);
- Each cell has a neighborhood of other cells. Altogether these form a network of cells, which is usually taken to be regular (e.g. the cells are arranged as a lattice with or without periodic boundary conditions);
- The evolution of all of the cells has a number of physics-like symmetries. Locality is one: the next state of a cell depends only on its current state and that of its neighbours. Homogeneity is another: the evolution acts the same everywhere, and is independent of time;
- The state space of the cells, and the operations performed on them, should be motivated by principles of quantum mechanics.

One feature that is often considered important for a model of quantum cellular automata is that it should be universal for quantum computation (i.e. that it can efficiently simulate quantum Turing machines,[1] [2] some arbitrary quantum circuit[3] or simply all other quantum cellular automata[4] [5] . Models which have been proposed recently impose further conditions, e.g. that quantum cellular automata should be reversible and/or local unitary, and have an easily determined global transition function from the rule for updating individual cells.[2] Recent results show that these properties can be derived axiomatically, from the symmetries of the global evolution.[6] [7] [8]

Models of QCA

Early proposals

Richard Feynman suggested an initial approach to quantizing a model of cellular automata[9] Gerhard Grössing and Anton Zeilinger introduced the term "quantum cellular automata" to refer to a model they defined in 1988:[10] however, their model has very little in common with the concepts developed in quantum computation after David Deutsch's formal development of that subject from 1985,[11] and so has not been developed significantly as a model of computation.

Models of universal quantum computation

The first formal model of quantum cellular automata to be researched in depth was that introduced by John Watrous.[1] This model was developed further by Wim van Dam,[12] as well as Christoph Dürr, Huong LêThanh, and Miklos Santha,[13] [14] Jozef Gruska.[15] and Pablo Arrighi.[16] However it was later realised that this definition was too loose, in the sense that some instances of it allow superluminal signalling.[6] [7] A second wave of models includes those of Susanne Richter and Reinhard Werner,[17] of Benjamin Schumacher and Reinhard Werner,[6] of Carlos Pérez-Delgado and Donny Cheung,[2] and of Pablo Arrighi, Vincent Nesme and Reinhard Werner.[7] [8] These are all closely related, and do not suffer any such locality issue. In the end one can say that they all agree to picture quantum cellular automata as just some large quantum circuit, infinitely repeating across time and space.

Models of physical systems

Models of quantum cellular automata have been proposed by David Meyer,[18] [19] by Bruce Bogosian and Washington Taylor,[20] and by Peter Love and Bruce Bogosian[21] as a means of simulating quantum lattice gases, motivated by the use of "classical" cellular automata to model classical physical phenomena such as gas dispersion.[22]

Quantum dot cellular automata

A proposal for implementing *classical* cellular automata by systems designed with quantum dots has been proposed under the name "quantum cellular automata" by Doug Tougaw and Craig Lent,[23] as a replacement for classical computation using CMOS technology. In order to better differentiate between this proposal and models of cellular automata which perform quantum computation, many authors working on this subject now refer to this as a quantum dot cellular automaton.

References

[1] Watrous, John (1995), "On one-dimensional quantum cellular automata", *Proc. 36th Annual Symposium on Foundations of Computer Science (Milwaukee, WI, 1995)*, Los Alamitos, CA: IEEE Comput. Soc. Press, pp. 528–537, doi:10.1109/SFCS.1995.492583, MR1619103.

[2] C. Pérez-Delgado and D. Cheung, "Local Unitary Quantum Cellular Automata", Phys. Rev. A 76, 032320, 2007. See also arXiv:0709.0006 (quant-ph) (http://www.arxiv.org/abs/0709.0006)

[3] D.J. Shepherd, T. Franz, R.F. Werner: Universally programmable Quantum Cellular Automaton. Phys. Rev. Lett. 97, 020502 (2006)

[4] P. Arrighi, R. Fargetton, Z. Wang, Intrinsically universal one-dimensional quantum cellular automata in two flavours, Fundamenta Informaticae Vol.91, No.2, pp.197-230, (2009). See also (quant-ph) (http://arxiv.org/abs/0704.3961)

[5] P. Arrighi, J. Grattage, A quantum Game of Life, Proceedings of JAC 2010, Turku, December 2010. TUCS Lecture Notes 13, 31-42, (2010). See also (quant-ph) (http://arxiv.org/abs/arXiv:1010.3120) and (Companion Website) (http://www.grattage.co.uk/jon/3DQCA)

[6] B. Schumacher and R. Werner, "Reversible quantum cellular automata", quant-ph/0405174 (http://www.arxiv.org/abs/quant-ph/0405174)

[7] Pablo Arrighi, Vincent Nesme, Reinhard Werner, One-dimensional quantum cellular automata over finite, unbounded configurations. See also (quant-ph) (http://arxiv.org/abs/0711.3517)

[8] Pablo Arrighi, Vincent Nesme, Reinhard Werner, N-dimensional quantum cellular automata. See also (quant-ph) (http://arxiv.org/abs/0711.3975)

[9] R. Feynman, "Simulating physics with computers", Int. J. Theor. Phys. **21**, 1982: pp. 467–488.

[10] G. Grossing and A. Zeilinger, "Quantum cellular automata", Complex Systems 2 (2), 1988: pp. 197–208 and 611–623

[11] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer" Proceedings of the Royal Society of London A 400 (1985), pp. 97–117.
[12] W. van Dam, "Quantum cellular automata", Master Thesis, Computer Science Nijmegen, Summer 1996.
[13] C. Dürr and M. Santha, "A decision procedure for unitary linear quantum cellular automata", quant-ph/9604007 (http://www.arxiv.org/abs/quant-ph/9604007).
[14] C. Dürr, H. LêTanh, M. Santha, "A decision procedure for well-formed linear quantum cellular automata", Rand. Struct. Algorithms 11, 1997: pp. 381–394. See also cs.DS/9906024 (http://www.arxiv.org/abs/cs.DS/9906024).
[15] J. Gruska, "Quantum Computing", McGraw-Hill, Cambridge 1999: Section 4.3.
[16] Pablo Arrighi, An algebraic study of unitary one dimensional quantum cellular automata, Proceedings of MFCS 2006, LNCS 4162, (2006), pp122-133. See also quant-ph/0512040 (http://arxiv.org/abs/quant-ph/0512040)
[17] S. Richter and R.F. Werner, "Ergodicity of quantum cellular automata", J. Stat. Phys. 82, 1996: pp. 963–998. See also cond-mat/9504001 (http://www.arxiv.org/abs/cond-mat/9504001)
[18] D. Meyer, "From quantum cellular automata to quantum lattice gases", Journal of Statistical Physics 85, 1996: pp. 551–574. See also quant-ph/9604003 (http://www.arxiv.org/abs/quant-ph/9604003).
[19] D. Meyer, "On the absence of homogeneous scalar unitary cellular automata'", Physics Letters A 223, 1996: pp. 337–340. See also quant-ph/9604011 (http://www.arxiv.org/abs/quant-ph/9604011).
[20] B. Bogosian and W. Taylor, "Quantum lattice-gas model for the many-particle Schrödinger equation in d dimensions", Physical Review E 57, 1998: pp. 54–66.
[21] P. Love and B. Bogosian, "From Dirac to Diffusion: Decoherence in Quantum Lattice Gases", Quantum Information Processing 4, 2005, pp. 335–354.
[22] B. Chophard and M. Droz, "Cellular Automata modeling of Physical Systems", Cambridge Universitry Press, 1998.
[23] P. Tougaw, C. Lent, "Logical devices implemented using quantum cellular automata", J. Appl. Phys. 75, 1994: pp. 1818–1825

Coupled map lattice

A **coupled map lattice (CML)** is a dynamical system that models the behavior of non-linear systems (especially partial differential equations). They are predominantly used to qualitatively study the chaotic dynamics of spatially extended systems. This includes the dynamics of spatiotemporal [1] chaos where the number of effective degrees of freedom diverge as the size of the system increases [2] . Features of the CML are discrete time dynamics, discrete underlying spaces (lattices or networks), and real (number or vector), local, continuous state variables [3] . Studied systems include populations, chemical reactions, convection, fluid flow and biological networks. Even recently, CMLs have been applied to computational networks [4] identifying detrimental attack methods and cascading failures.

CML's are comparable to cellular automata models in terms of their discrete features [5] . However, the value of each site in a cellular automata network is strictly dependent on its neighbor (s) from the previous time step. Each site of the CML is only dependent upon its neighbors relative to the coupling term in the recurrence equation. However, the similarities can be compounded when considering multicomponent dynamical systems.

Introduction

The modeling of a CML generally incorporates a system of equations (coupled or uncoupled), a finite number of variables, a global or local coupling scheme and the corresponding coupling terms. The dimension of the underlying lattice can exist in infinite dimensions, but for this observation we restrict the lattice to two. Mappings of interest in CMLs generally demonstrate a chaotic behavior. Such maps can be found here: List of chaotic maps.

A logistic mapping demonstrates chaotic behavior, easily identifiable in one dimension for parameter r > 3.57 (see Logistic map). It is graphed across a small lattice and decoupled with respect to neighboring sites. The recurrence equation is homogeneous, albeit randomly seeded. The parameter r is updated every time step (see Figure 1, Enlarge, Summary):

$$x_{n+1} = r x_n (1 - x_n)$$

The result is a raw form of chaotic behavior in a map lattice. The range of the function is bounded so similar contours through the lattice is expected. However, there are no significant spatial correlations or pertinent fronts to the chaotic behavior. No obvious order is apparent.

For a basic coupling, we consider a 'single neighbor' coupling where the value at any given site s is mapped recursively with respect to itself and the neighboring site $s-1$. The coupling parameter $\epsilon = 0.5$ is equally weighted.

$$x_{n+1} = (\epsilon)[rx_n(1-x_n)]_s + (1-\epsilon)[rx_n(1-x_n)]_{s-1}$$

Even though each native recursion is chaotic, a more solid form develops in the evolution. Elongated convective spaces persist throughout the lattice (see Figure 2).

Figure 1: An uncoupled logistic map lattice with random seeding over forty iterates.

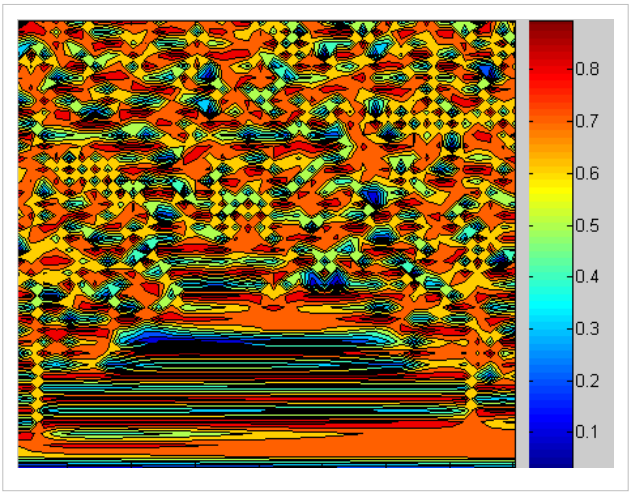


Figure 2: A CML with a single-neighbor coupling scheme taken over forty iterates.

History

CMLs were first introduced in the mid 1980's through a series of closely released publications [6] [7] [8] [9] . Kapral used CMLs for modeling chemical spatial phenomena. Kuznetsov sought to apply CMLs to electrical circuitry by developing a renormalization group approach (similar to Feigenbaum's universality to spatially extended systems). Kaneko's focus was more broad and he is still known as the most active researcher in this area [10] .The most examined CML model was introduced by Kaneko in 1983 where the recurrence equation is as follows:

$$u_s^{t+1} = (1-\varepsilon)f(u_s^t) + \frac{\varepsilon}{2}\left(f(u_{s+1}^t) + f(u_{s-1}^t)\right) \quad t \in \mathbb{N},\ \varepsilon \in [0,1]$$

where $u_s^t \in \mathbb{R}$,and f is a real mapping.

The applied CML strategy was as follows:

- Choose a set of field variables on the lattice at a macroscopic level. The dimension (not limited by the CML system) should be chosen to correspond to the physical space being researched.
- Decompose the process (underlying the phenomena) into independent components.
- Replace each component by a nonlinear transformation of field variables on each lattice point and the coupling term on suitable, chosen neighbors.
- Carry out each unit dynamics ("procedure") successively.

Classification

The CML system evolves through discrete time by a mapping on vector sequences. These mappings are a recursive function of two competing terms: an individual nonlinear reaction, and a spatial interaction (coupling) of variable intensity. CMLs can be classified by the strength of this coupling parameter(s).

Much of the current published work in CMLs is based in weak coupled systems [11] where diffeomorphism of the state space close to identity are studied. Weak coupling with monotonic (bistable) dynamical regimes demonstrate spatial chaos phenomena and are popular in neural models [12] . Weak coupling unimodal maps are characterized by their stable periodic points and are used by genetic regulatory network models. Space-time chaotic phenomena can be demonstrated from chaotic mappings subject to weak coupling coefficients and are popular in phase transition phenomena models.

Intermediate and strong coupling interactions are less prolific areas of study. Intermediate interactions are studied with respect to fronts and traveling waves, riddled basins, riddled bifurcations, clusters and non-unique phases. Strong coupling interactions are most well known to model synchronization effects of dynamic spatial systems such as the Kuramoto model.

These classifications do not reflect the local or global (GMLs [13])coupling nature of the interaction. Nor do they consider the frequency of the coupling which can exist as a degree of freedom in the system [14] . Finally, they do not distinguish between sizes of the underlying space or boundary conditions.

Surprisingly the dynamics of CMLs have little to do with the local maps that constitute their elementary components. With each model a rigorous mathematical investigation is needed to identify a chaotic state (beyond visual interpretation). Rigorous proofs have been performed to this effect. By example: the existence of space-time chaos in weak space interactions of one-dimensional maps with strong statistical properties was proven by Bunimovich and Sinai in 1988 [15] . Similar proofs exist for weakly hyperbolic maps under the same conditions.

Unique CML qualitative classes

CMLs have revealed novel qualitative universality classes in (CML) phenomenology. Such classes include:

- Spatial bifurcation and frozen chaos
- Pattern Selection
- Selection of zig-zag patterns and chaotic diffusion of defects
- Spatio-temporal intermittency
- Soliton turbulence
- Global traveling waves generated by local phase slips
- Spatial bifurcation to down-flow in open flow systems.

Visual phenomena

The unique qualitative classes listed above can be visualized. By applying the Kaneko 1983 model to the logistic $f(x_n) = 1 - ax^2$ map, several of the CML qualitative classes may be observed. These are demonstrated below, note the unique parameters:

Frozen Chaos	**Pattern Selection**	**Chaotic Brownian Motion of Defect**

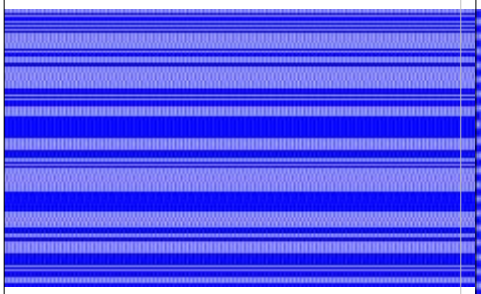
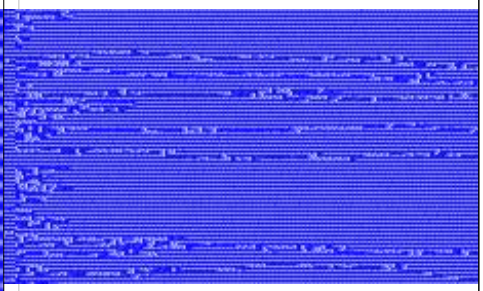

		

Figure 1: Sites are divided into non-uniform clusters, where the divided patterns are regarded as attractors. Sensitivity to initial conditions exist relative to $a < 1.5$.	Figure 2: Near uniform sized clusters ($a = 1.71$, $\varepsilon = 0.4$).	Figure 3: Deflects exist in the system and fluctuate chaotically akin to Brownian motion ($a = 1.85$, $\varepsilon = 0.1$).
Defect Turbulence	**Spatiotemporal Intermittency I**	**Spatiotemporal Intermittency II**

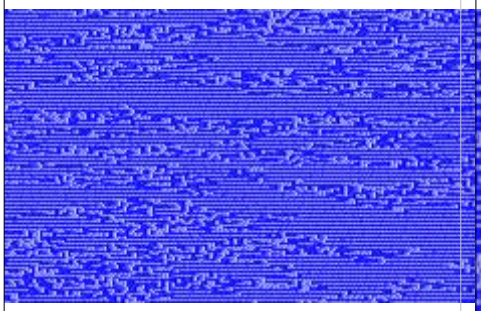
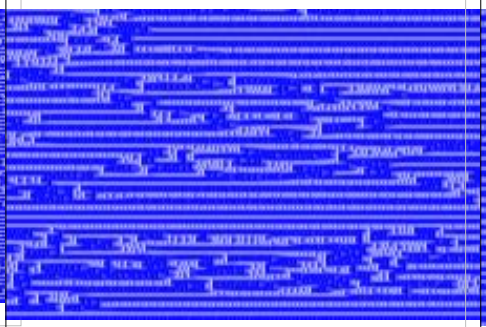
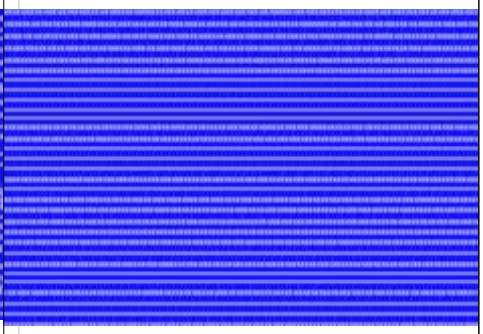

		

Figure 4: Many defects are generated and turbulently collide ($a = 1.895$, $\varepsilon = 0.1$).	Figure 5: Each site transits between a coherent state and chaotic state intermittently ($a = 1.75$, $\varepsilon = 0.6$), Phase I.	Figure 6: The coherent state, Phase II.
Fully Developed Spatiotemporal Chaos	**Traveling Wave**	

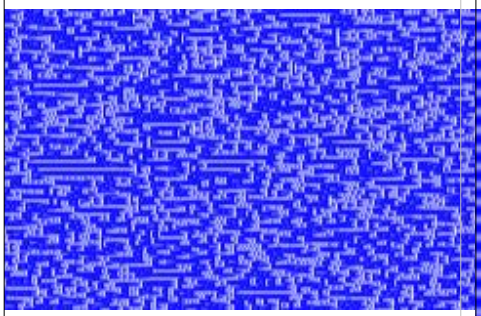



Figure 7: Most sites independently oscillate chaotically ($a = 2.00$, $\varepsilon = 0.3$).	Figure 8: The wave of clusters travels at 'low' speeds ($a = 1.47$, $\varepsilon = 0.5$).	

Quantitative analysis quantifiers

Coupled map lattices being a prototype of spatially extended systems easy to simulate have represented a benchmark for the definition and introduction of many indicators of spatio-temporal chaos, the most relevant ones are

- The power spectrum in space and time
- Lyapunov spectra[16]
- Dimension density
- Kolmogorov–Sinai entropy density
- Distributions of patterns
- Pattern entropy
- Propagation speed of finite and infinitesimal disturbance
- Mutual information and correlation in space-time
- Lyapunov exponents, localization of Lyapunov vectors
- Comoving and sub-space time Lyapunov exponents.
- Spatial and temporal Lyapunov exponents [17]

References

[1] http://en.wiktionary.org/wiki/spatiotemporal
[2] Kaneko, Kunihiko. "Overview of Coupled Map Lattices." Chaos 2, Num3(1992): 279.
[3] Chazottes, Jean-René, and Bastien Fernandez. Dynamics of Coupled Map Lattices and of Related Spatially Extended Systems. Springer, 2004. pgs 1–4
[4] Xu, Jian. Wang, Xioa Fan. " Cascading failures in scale-free coupled map lattices." IEEE International Symposium on Circuits and Systems " ISCAS Volume 4, (2005): 3395–3398.
[5] R. Badii and A. Politi, Complexity: Hierarchical Structures and Scaling in Physics (Cambridge University Press,Cambridge, England, 1997).
[6] K. Kaneko, Prog. Theor. Phys. 72, 480 (1984)
[7] I. waller and R. Kapral; Phys. Rev. A 30 2047 (1984)
[8] J. Crutchfield, Phyisca D 10, 229 (1984)
[9] S. P.Kuznetsov and A. S. Pikovsky, Izvestija VUS, Radiofizika 28, 308 (1985)
[10] http://chaos.c.u-tokyo.ac.jp/
[11] Lectures from the school-forum (CML 2004) held in Paris, June 21{July 2, 2004. Edited by J.-R. Chazottes and B. Fernandez. Lecture Notes in Physics, 671. Springer, Berlin (2005)
[12] Nozawa, Hiroshi. "A neural network model." Chaos 2, Num3(1992): 377.
[13] Ho, Ming-Ching. Hung, Yao-Chen. Jiang, I-Min. "Phase synchronization in inhomogenous globally coupled map lattices. Physics Letter A. 324 (2004) 450–457. (http://www.phys.sinica.edu.tw/~statphys/publications/2004_full_text/M_C_Ho_PLA_324_450(2004).pdf)
[14] http://www.mat.uniroma2.it/~liverani/Lavori/live0803.pdf
[15] L.A. Bunimovich and Ya. G. Sinai. "Nonlinearity" Vol. 1 pg 491 (1988)
[16] Lyapunov Spectra of Coupled Map Lattices, S. Isola, A. Politi, S. Ruffo, and A. Torcini (http://www.fi.isc.cnr.it/users/antonio.politi/Reprints/052.pdf)
[17] S. Lepri, A. Politi and A. Torcini Chronotopic Lyapunov Analysis: (I) a Detailed Characterization of 1D Systems (http://xxx.lanl.gov/abs/chao-dyn/9504005), J. Stat. Phys., 82 5/6 (1996) 1429.

Further reading

- Google Library (2005). *Dynamics of Coupled Map Lattices* (http://books.google.com/books?id=a63Q8DhKA44C&dq=coupled+map+lattices&source=gbs_summary_s&hl=en). Springer. ISBN 9783540242895. Archived from the original (http://books.google.com/?id=a63Q8DhKA44C&dq=coupled+map+lattices) on 2008-03-29.
- Shawn D. Pethel, Ned J. Corron, and Erik Bollt. "Symbolic Dynamics of Coupled Map Lattices" (http://people.clarkson.edu/~bolltem/Papers/PhysRevLett_96_034105PethelCorronBollt.pdf). Physical Review Letters. Archived from the original (http://dx.dio.org/10.1103/PhysRevLett.96.034105) on 2008-03-29.
- E. Atlee Jackson, *Perspectives of Nonlinear Dynamics: Volume 2* (http://books.google.com/books?id=M2E0AAAAIAAJ&source=gbs_ViewAPI), Cambridge University Press, 1991, ISBN 0521426332

- H.G, Schuster and W. Just, *Deterministic Chaos* (http://www.whsmith.co.uk/CatalogAndSearch/ProductDetails.aspx?productID=9783527404155), John Wiley and Sons Ltd, 2005, ISBN 3527404155
- Introduction to Chaos and Nonlinear Dynamics (http://brain.cc.kogakuin.ac.jp/~kanamaru/Chaos/e/)

External links

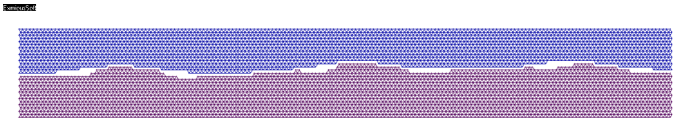
- Kaneko Laboratory (http://chaos.c.u-tokyo.ac.jp/)
- Institut Henri Poincaré, Paris, June 21 – July 2, 2004 (http://www.cpht.polytechnique.fr/cpth/cml2004/)
- Istituto dei Sistemi Complessi (http://www.fi.isc.cnr.it/), Florence, Italy

Software

- Java CML/GML web-app (http://brain.cc.kogakuin.ac.jp/~kanamaru/Chaos/e/CMLGCM/)
- AnT 4.669 – A simulation and Analysis Tool for Dynamical Systems (http://ant4669.de/)

Movable cellular automaton

Movable cellular automaton method

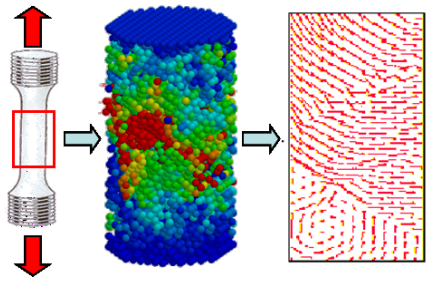


Movable cellular automaton actively changed self neighbors by means switching neighbors state from linked to unlinked and vice versa (modeling of contact interaction)

Method type	
Continuous/Discrete	Discrete
Analytical/Computational	Computational
Characteristics	
Influenced by	cellular automaton, discrete element
Method in	computational solid mechanics

The **Movable cellular automaton (MCA)** method is a method in computational solid mechanics based on the discrete concept. It provides advantages both of classical cellular automaton and discrete element methods. Important advantage of the MCA method is a possibility of direct simulation of materials fracture including damage generation, crack propagation, fragmentation and mass mixing. It is difficult to simulate these processes by means of continuum mechanics methods (For example: finite element method, finite difference method, etc.), so some new concepts like peridynamics is required. Discrete element method is very effective to simulate granular materials, but mutual forces among movable cellular automata provides simulating solids behavior. If size of automaton will be close to zero then MCA behavior becomes like classical continuum mechanics methods.

Keystone of the movable cellular automaton method

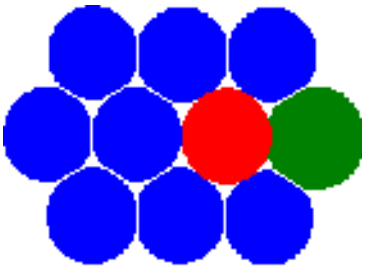


Object (at left) is described as set of interacted automata (at center). At right is shown velocity field of automata.

In framework of the **MCA** approach an object under modeling is considered as a set of interacting elements/automata. The dynamics of the set of automata are defined by their mutual forces and rules for their relationships. This system exists and operates in time and space. Its evolution in time and space is governed by the equations of motion. The mutual forces and rules for inter-elements relationships are defined by the function of the automaton response. This function has to be specified for each automaton. Due to mobility of automata the following new parameters of cellular automata have to be included into consideration: R^i – radius-vector of automaton; V^i – velocity of automaton; $\boldsymbol{\omega}^{\mathbf{i}}$ – rotation velocity of automaton; θ^i – rotation vector of automaton; m^i – mass of automaton; J^i – moment of inertia of automaton.

New concept: neighbours

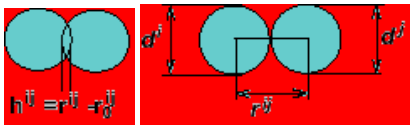
The new concept of the MCA method is based on the introducing of the **state of the pair of automata** (relation of interacting pairs of automata) in addition to the conventional one – the state of a separate automaton. Note that the introduction of this definition allows to go from the static net concept to the **concept of neighbours**. As a result of this, the automata have the ability to change their neighbors by switching the states (relationships) of the pairs.



Each automaton has some neighbors

Definition of the parameter of pair state

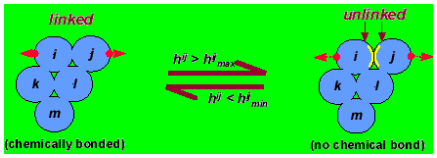
The introducing of new type of states leads to new parameter to use it as criteria for **switching relationships**. It is defined as an automaton overlapping parameters h^{ij}. So the relationship of the cellular automata is characterised by the value of their **overlapping**.



The initial structure is formed by setting up certain relationships among each pair of neighboring elements.

Criterion of switching of the state of pair relationships

In contrast to the classical cellular automaton method in the MCA method not only a single automaton but also a **relationship of pair of automata can be switched**. According with the bistable automata concept there are two types of the pair states (relationships):



At left pair of automata ij is linked. At right pair of automata ij is unlinked.

linked	– both automata belong to a solid
unlinked	– each automaton of the pair belongs to different bodies or parts of damaged body.

So the **changing of the state of pair relationships** is controlled by relative movements of the automata and the media formed by such pairs can be considered as bistable media.

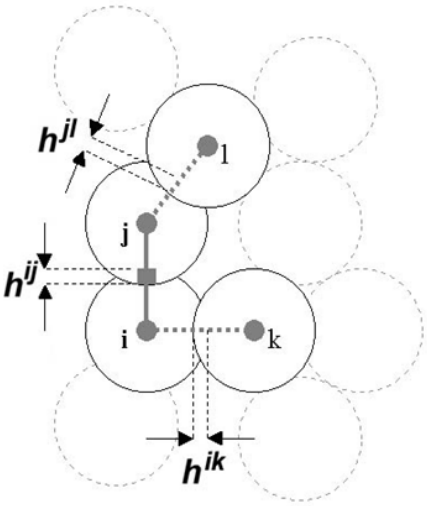
Equations of MCA motion

The evolution of MCA media is described by the following **equations of motion for translation**:

$$\frac{d^2h^{ij}}{dt^2} = \left(\frac{1}{m^i} + \frac{1}{m^j}\right) p^{ij} + \sum_{k \neq j} C(ij,ik)\psi(\alpha_{ij,ik})\frac{1}{m^i}p^{ik} + \sum_{l \neq i} C(ij,jl)\psi(\alpha_{ij,jl})\frac{1}{m^j}p^{jl}$$

Here m^i is the mass of automaton i, p^{ij} is central force acting between automata i and j, C(ij,ik) is certain coefficient associated with transferring the h parameter from pair **ij** to pair **ik**, $\psi(\alpha_{ij,ik})$ is angle between directions **ij** and **ik**.

Due to finite size of movable automata the rotation effects have to be taken into account. The **equations of motion for rotation** can be written as follows:



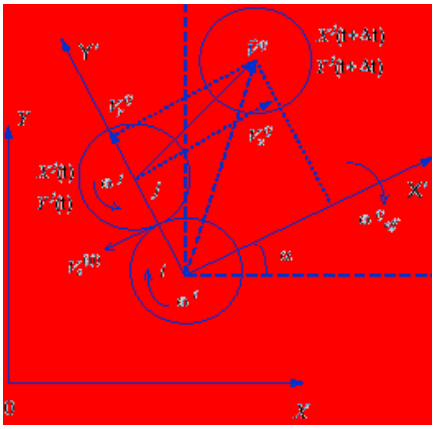
Forces between automata ij coming from their neighbors.

$$\frac{d^2\theta^{ij}}{dt^2} = \left(\frac{q^{ij}}{J^i} + \frac{q^{ji}}{J^j}\right)\tau^{ij} + \sum_{k \neq j} S(ij,ik)\frac{q^{ik}}{J^i}\tau^{ik} + \sum_{l \neq j} S(ij,jl)\frac{q^{jl}}{J^j}\tau^{jl}$$

Here Θ^{ij} is the angle of relative rotation (it is a switching parameter like h^{ij} for translation), q^{ij} is the distance from center of automaton **i** to contact point of automaton **j** (moment arm), τ^{ij} is the pair tangential interaction, S(ij,ik) is certain coefficient associated with transferring the Θ parameter from one pair to other (it is similar to C(ij,ik) from the equation for translation).

It should be noted that these equations are completely similar to the equations of motion for the many–particle approach.

Definition of deformation in pair of automata



Rotation of body as whole not caused to deformation in pair of automata

Translation of the pair automata The dimensionless deformation parameter for translation of the **i j** automata pair can be presented as:

$$\varepsilon^{ij} = \frac{h^{ij}}{r_0^{ij}} = \frac{\left(q^{ij} + q^{ji}\right) - \left(d^i + d^j\right)/2}{\left(d^i + d^j\right)/2}$$

In this case:

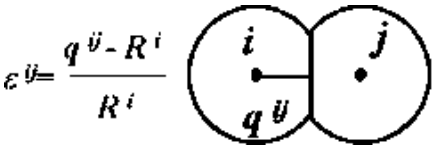
$$\left(\Delta\varepsilon^{i(j)} + \Delta\varepsilon^{j(i)}\right)\frac{\left(d^i + d^j\right)}{2} = V_n^{ij}\Delta t$$

where **Δt** time step, $\mathbf{V_n^{ij}}$ – relative velocity.

Rotation of the pair automata can be calculated by analogy with the last translation relationships.

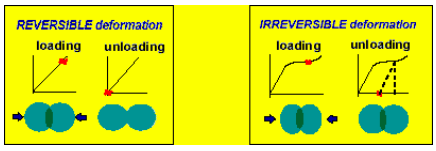
Modeling of irreversible deformation in the MCA method

The $\boldsymbol{\varepsilon^{ij}}$ parameter is used as a measure of deformation of automaton **i** under its interaction with automaton **j**. Where $\mathbf{q^{ij}}$ – is a distance from the center of automaton **i** to its contact point with automaton **j**; $\mathbf{R^i = d^i/2}$ ($\mathbf{d^i}$ – is the size of automaton **i**).

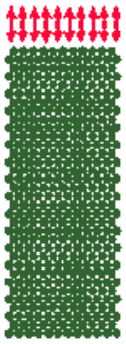
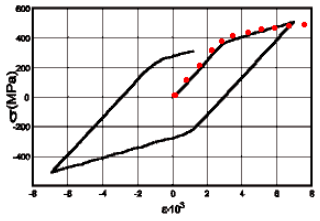


Deformation is determine by value of distance from the center of automaton

As an example the titanium specimen under cyclic loading (tension – compression) is considered. The loading diagram is shown in the next figure:



There are two types of the response function of automata

Scheme of loading	Loading diagram
	
	(**Red marks** are the experimental data)

Advantages of MCA method

Due to mobility of each automaton the MCA method allows to take into account directly such actions as:

- mass mixing
- penetration effects
- chemical reactions
- intensive deformation
- phase transformations
- accumulation of damages
- fragmentation and fracture
- cracks generation and development

Using boundary conditions of different types (fixed, elastic, viscous-elastic, etc.) it is possible to imitate different properties of surrounding medium, containing the simulated system. It is possible to model different modes of mechanical loading (tension, compression, shear strain, etc.) by setting up additional conditions at the boundaries.

References

- Psakhie, S.G.; Horie, Y.; Korostelev, S.Yu.; Smolin, A.Yu.; Dmitriev, A.I.; Shilko, E.V.; Alekseev, S.V. (November. - 1995). "Method of movable cellular automata as a tool for simulation within the framework of mesomechanics". *Russian Physics Journal* (Springer New York) **38** (11): 1157–1168. doi:10.1007/BF00559396.
- Psakhie, S.G.; Korostelev, S.Y.; Smolin, A.Y.; Dmitriev, A.I.; Shilko, E.V.; Moiseenko D.D.; Tatarincev E.M.; Alekseev, S.V. (1998). "Movable cellular automata method as a tool for physical mesomechanics of materials". *Physical mesomechanics* (The Organization of the Russian Academy of Sciences, Institute of Strength Physics and Materials Science of the Siberian Branch of RAS (ISPMS SB RAS)) **1** (1): 95–108. (Russian: Псахье, С.Г.; Коростелев, С.Ю.; Смолин, А.Ю.; Дмитриев, А.И.; Шилько, Е.В.; Моисеенко, Д.Д.; Татаринцев, Е.М.; Алексеев, С.В. (1998). "Метод подвижных клеточных автоматов как инструмент физической мезомеханики материалов" [1]. *Физическая мезомеханика* (Учреждение Российской академии наук Институт физики прочности и материаловедения Сибирского отделения РАН (ИФПМ СО РАН)) **1** (1): 95–108. Retrieved 2010-03-03.)
- Psakhie, S.G.; Ostermeyer, G.P.; Dmitriev, A.I.; Shilko, E.V.; Smolin, A.Y.; Korostelev, S.Y. (2000). "Movable cellular automata method as new direction in numerical discrete mechanics. I. Theoretical description". *Physical mesomechanics* (The Organization of the Russian Academy of Sciences, Institute of Strength Physics and Materials Science of the Siberian Branch of RAS (ISPMS SB RAS)) **3** (2): 5–13. (Russian: Псахье, С.Г.; Остермайер, Г.П.; Дмитриев, А.И.; Шилько, Е.В.; Смолин, А.Ю.; Коростелев, С.Ю. (2000). "Метод подвижных клеточных автоматов как новое направление дискретной вычислительной механики. I. Теоретическое описание" [2]. *Физическая мезомеханика* (Учреждение Российской академии наук Институт физики прочности и материаловедения Сибирского отделения РАН (ИФПМ СО РАН)) **3** (2): 5–13. Retrieved 2010-03-03.)
- Psakhie, S.G.; Horie, Y.; Ostermeyer, G.P.; Korostelev, S.Yu.; Smolin, A.Yu.; Shilko, E.V.; Dmitriev, A.I.; Blatnik, S.; Spegel, M.; Zavsek, S. (December 2001). "Movable cellular automata method for simulating materials with mesostructure" [3]. *Theoretical and Applied Fracture Mechanics* (Elsevier Science Ltd.) **37** (1-3): 311–334. doi:10.1016/S0167-8442(01)00079-9.
- Psakhie, S.G.; Smolin, A.Y.; Stefanov, Y.P.; Makarov, P.V.; Chertov, M.A. (2004). "Modeling the behavior of complex media by jointly using discrete and continuum approaches". *Technical Physics Letters* (MAIK Nauka/Interperiodica distributed exclusively by Springer Science+Business Media LLC.) **30** (9): 712–714. doi:10.1134/1.1804572.
- Shimizu, Y.; Hart, R.; Cundall, P. (2004). *Numerical modeling in Micromechanics via Particle Methods* [4]. ISBN 9058096793. Retrieved 2010-03-03.

- Gnecco, E.; Meyer E. (Eds.) (2007). *Fundamentals of friction and wear on the Nanoscale* [5]. ISBN 9783540368069. Retrieved 2010-03-03.
- Yunliang, Tan; Guirong, Teng; Haitao, Li. "MCA Model for Simulating the Failure of Microinhomogeneous Materials". *Journal of Nanomaterials* (Hindawi Publishing Corporation) **2008**: 1–7. doi:10.1155/2008/946038. 946038.
- Fomin, V.M.; Andreev, A.N. and al (2008). *Mechanics - from discrete to continious*. Russian academy of science, Siberian branch, Institute of theoretical and applied mechanics (named S.A. Khristianovich). p. 344. ISBN 978-5-7692-0974-1. (Russian: Фомин, В.М.; Андреев А.Н. и др. (2008). *Механика - от дискретного к сплошному* [6]. Рос. акад наук, Сиб. отд-ние, Ин-т теоретической и прикладной механики им. С.А. Христиановича. p. 344. ISBN 978-5-7692-0974-1. Retrieved 3.03.2010.)
- Smolin, A.Y.; Roman, N.V.; Dobrynin, S.A.; Psakhie, S.G. (May-August. - 2009). "On rotation in the movable cellular automaton method". *Physical Mesomechanics* (Elsevier B.V.) **12** (3-4): 124–129. doi:10.1016/j.physme.2009.07.004.
- Popov, Valentin L. (2009). *Kontaktmechanik und Reibung (Ein Lehr- und Anwendungsbuch von der Nanotribologie bis zur numerischen Simulation)*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-88837-6. ISBN 9783540888369.
- Dobrynin, S.A. (2010). *Development of movable cellular atomaton method for modeling generation and propagation of elastic waves under contact interaction of solids*. Tomsk: PhD thesis in physics and mathematics. p. 130. (Russian: Добрынин, С.А. (2010). *Развитие метода подвижных клеточных автоматов для моделирования генерации и распространения упругих волн при контактном взаимодействии твердых тел* [7]. Томск: Диссертация … кандидата физико-математических наук. p. 130. Retrieved 3.03.2010.)

Software

- MCA software package [8]
- Software for simulation of materials in discrete-continuous approach «FEM+MCA»: Number of state registration in Applied Research Foundation of Algorithms and Software (AFAS): 50208802297 / Smolin A.Y., Zelepugin S.A., Dobrynin S.A.; applicant and development center is Tomsk State University. – register date 28.11.2008; certificate AFAS N 11826 date 01.12.2008. [9]

References

[1] http://elibrary.ru/item.asp?id=12913617
[2] http://elibrary.ru/item.asp?id=12913646
[3] http://mechanik.tu-berlin.de/popov/software/mca/TAFMEC.pdf
[4] http://books.google.com/books?id=GVAUH98Yr6oC&lpg=PA242&dq=%22Movable%20cellular%20automaton%22%20-inpublisher%3Aicon&as_brr=0&hl=ru&pg=PR4#v=onepage&q=%22Movable%20cellular%20automaton%22%20-inpublisher:icon&f=false
[5] http://books.google.com/books?id=v2Pe5thhNiwC&lpg=PP1&hl=ru&pg=PP1#v=onepage&q=&f=false
[6] http://www.sibran.ru/psb/books/show_text.phtml?rus+1880+psb
[7] http://serg-dobrinin.narod.ru/disert/index.html
[8] http://mechanik.tu-berlin.de/popov/software/mca/mca.htm
[9] http://serg-dobrinin.narod.ru/mydiploms/reg_program.jpg

Article Sources and Contributors

Cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=451448848 *Contributors*: -Ril-, 524, ACW, Acidburn24m, AdRock, Agora2010, Akramm1, Alexwg, Allister MacLeod, Alpha Omicron, Angela, AnonEMouse, Anonymous Dissident, Argon233, Asmeurer, Axd, AxelBoldt, B.huseini, Baccyak4H, Balsarxml, Banus, Bearian, Beddowve, Beeblebrox, Benjah-bmm27, Bento00, Bevo, Bhumiya, BorysB, Bprentice, Brain, Bryan Derksen, Caileagleisg, Calwiki, CharlesC, Chmod007, Chopchopwhitey, Christian Kreibich, Chuckwolber, Ckatz, Crazilla, Cstheoryguy, Curps, DVdm, Dalf, Dave Feldman, David Eppstein, Dawnseeker2000, Dcornforth, Deltabeignet, Dhushara, Dmcq, Dra, Dysprosia, EagleFan, Edward Z. Yang, Elektron, EmreDuran, Erauch, Eric119, Error, Evil saltine, Ezubaric, Felicity Knife, Ferkel, FerrenMacI, Froese, GSM83, Geneffects, Giftlite, Gioto, Gleishma, Gragus, Graham87, GregorB, Gthen, Guanaco, HairyFotr, Hannes Eder, Headbomb, Hephaestos, Hfastedge, Hillgentleman, Hiner, Hmonroe, Hope09, I do not exist, Ideogram, Ilmari Karonen, Imroy, InverseHypercube, Iridescent, Iseeaboar, Iztok.jeras, J.delanoy, JaGa, Jasper Chua, Jdandr2, Jlopez1967, JocK, Joeyramoney, Jogloran, Jon Awbrey, Jonkerz, Jose Icaza, Joseph Myers, JuliusCarver, Justin W Smith, K-UNIT, Kaini, Karlscherer3, Kb, Keenan Pepper, Kiefer.Wolfowitz, Kieff, Kizor, Kku, Kneb, Kotasik, Kyber, Kzollman, LC, Laesod, Lamro, Lgallindo, Lightmouse, Lpdurocher, LunchboxGuy, Mandalaschmandala, MarSch, Marasmusine, Marcus Wilkinson, Mattisse, Mbaudier, Metric, Mgiganteus1, Michael Hardy, Mihai Damian, Mosiah, MrOllie, Mudd1, MuthuKutty, Mydogtrouble, NAHID, Nakon, Nekura, Ninly, Oliviersc2, On you again, Orborde, Oubiwann, P0lyglut, PEHowland, Pcorteen, Peak Freak, Perceval, Phaedriel, Pi is 3.14159, PierreAbbat, Pixelface, Pleasantville, Pygy, Quuxplusone, R.e.s., RDBury, Radagast83, Raven4x4x, Requestion, RexNL, Rjwilmsi, Robin klein, RyanB88, Sadi Carnot, Sam Tobar, Samohyl Jan, Sbp, ScAvenger, Schneelocke, Selket, Setoodehs, Shoemaker's Holiday, Smjg, Spectrogram, Srleffler, Sumanafsu, SunCreator, Svrist, The Temple Of Chuck Norris, Throwaway85, Tijfo098, TittoAssini, Tobias Bergemann, Torcini, Tropylium, Ummit, Versus22, Visor, Warrado, Watcher, Watertree, Wavelength, Welsh, Wik, William R. Buckley, Wolfpax50, XJamRastafire, Xerophytes, Xihr, Yonkeltron, Yugsdrawkcabeht, ZeroOne, Zoicon5, Zom-B, Zorbid, 329 anonymous edits

Brian's Brain *Source*: http://en.wikipedia.org/w/index.php?oldid=367396452 *Contributors*: Andrewjlockley, Colfulus, Ferkel, Ken g6, PamD, Simpsons contributor, Zodon, 3 anonymous edits

Conway's Game of Life *Source*: http://en.wikipedia.org/w/index.php?oldid=455327383 *Contributors*: 213.253.40.xxx, ABoerma, ACW, Abelson, Adijk, Ae-a, Aesir, Agorf, Alessio88, AlphaPyro, Amakuru, Amux, Andy120290, Andykt, Anomalocaris, AnonMoos, AnthonyQBachler, Aplusbi, Arbol01, Ariel., Arto B, Ashley Pomeroy, Atakdoug, AugPi, Awaterl, B1u SkR33N, Baccyak4H, Baltar, Gaius, Banno, Betacommand, Bevo, Bitbutter, Bkell, Bloodrage, BooHamster, Bookandcoffee, BookgirlST, Borisblue, Bryan.burgers, Bubba73, Bud Charles, Butros, Cabbers, Caekaert, Calcyman, CanisRufus, Capricorn42, Captain-n00dle, Carey Evans, Charles Matthews, Chris 73, Christopher.Madsen, Chuck Smith, Clarince63, Closedmouth, D6, DFRussia, DIG, DMacks, DNewhall, DV8 2XL, DaGizza, Daelin, Damian Yerrick, Darth Mike, Darth Panda, Darthrevan1789, David Eppstein, DavidWBrooks, Davipo, Dclayh, Deadlyfish, Delldot, Diego, Dimitrii, Discospinster, Dispenser, Dmitry123456, Doradus, Dori, DragonflySixtyseven, Ds13, Dvgrn, Dvorak729, Dxco, Dylan Lake, Dysprosia, E-trail, ESkog, Emurphy42, Eric119, Esger, EverGreg, Evercat, Faisal.akeel, Ffransoo, FiP, Flewis, Fprefect111, Frecklefoot, Fredrik, Froese, Gaius Cornelius, Gdr, Geneffects, Gerbrant, Gevil, Giftlite, Gioto, Gleishma, Gorank4, Graham87, Graue, GregorB, Grontesca, Gulliveig, Gwern, Headbomb, Heliac, Henrygb, HexaChord, Hillgentleman, Home Row Keysplurge, Hughsk, Hyperdeath, Ihope127, Immunize, IntrigueBlue, Ioscius, Ipso2, Ixfd64, JTN, JYOuyang, Jarl Friis, Jbragadeesh, Jean-claude perez, Jim.belk, Jleedev, Jlopez1967, Jmundo, Joee92, John.constantine, Johnf.at.home, Johnuniq, JokeySmurf, Jpo, Jrockley, Julescubtree, Justin W Smith, Jwz, Kaldari, Kappa, Katalaveno, KenWalker, Kenirwin, Kieff, Kim Bruning, Kosmar, Kubigula, Kurykh, KymFarnik, LC, LOL, La goutte de pluie, Lanthanum-138, LedgendGamer, LeonardoRob0t, Lesnail, Lesonyrra, Livajo, M.nelson, Markaci, Markhurd, Marktompsett, Martarius, Marudubshinki, Maury Markowitz, McGeddon, Meggar, Melesse, MeltBanana, Michael L. Kaufman, MisfitToys, MisterSanderson, Mitsukai, Mk5384, Mkill, Moondoll, Muriel Gottrop, Narumara, Nealmcb, NevilleDNZ, Nick Levine, Nikevich, Noe, Notheruser, NuclearCarnage, NuclearWarfare, Obrienmi8, Ogxela, Oliver Pereira, Opelio, Owain, P0lyglut, Pakaran, Palpalpalpal, Panoptical, Paul A, Paul August, Paul G, PauloCalipari, Pavel Jelinek, Pcr, Perl, Pharaoh of the Wizards, Phil Boswell, Pichote, Pizza1512, Plook, PuzzletChung, Python eggs, Quentin X, Quiddity, Quuxplusone, RDBury, Racklever, Raekwon, Randomphantom45, Rangek, Rat at WikiFur, Rbtmdl, Redslazer, RekishiEJ, Reonic, Retired username, Richard001, Robin klein, RodrigoCamargo, RogerBarnett, Rossumcapek, Rspeer, SDS, SGBailey, Sabbut, Salix alba, Sam Tobar, Sandman303, Satbir15, Savant13, ScAvenger, Sciguy47, Scote, Scuzzi, Shadowcheets, Shanes, Shellreef, Sheridan, Shizhao, Shoemaker's Holiday, Simpsons contributor, Sir Isaac, Sir Paul, Sjorford, Snoyes, Solipsist, Spellmaster, Srleffler, Standardfact, Stay cool, Steven Watson, Sverdrup, Tarquin, Tedweird, Tenbaset, The Anome, The Earwig, TheAllSeeingEye, Theusernameiwantedisalreadyinuse, Thumperward, Tim1988, Tkorrovi, Tom harrison, Tomyumgoong, Trovatore, Trubye, Turidoth, Twri, UBERNESS, Updatehelper, Usb10, UtherSRG, Val42, Versus22, Voidxor, Wavelength, Wayfarer, Weyes, Wiggy04, Wknight94, Wody1025, Xhin, Ynhockey, ZICO, Zdim, Zundark, Zvika, 435 anonymous edits

Langton's ant *Source*: http://en.wikipedia.org/w/index.php?oldid=450320214 *Contributors*: -Lord-92, Alpha Omicron, Altenmann, AnAj, Borislav, Bryan Derksen, Calcyman, Charles Matthews, Chas zzz brown, Cholling, CommonsDelinker, Conti, DataWraith, Dcoetzee, Derek Ross, Dnas, Dysprosia, E Wing, El C, Eric119, Ferkel, FrankBuss, FvdP, HairyFotr, Jamelan, Kotasik, Lesnail, Lumos3, MER-C, Matthew Stannard, Mcpchrupka, Michael Hardy, Oliver Pereira, Quintopia, RDBury, Rajah, Ruud Koot, Simeon, Simpsons contributor, Slakr, Sojournist, StealthFox, SvNH, The Anome, Thumperward, Tommy2010, Wikiskimmer, Wolfkeeper, Wyllyjon, 43 anonymous edits

Wireworld *Source*: http://en.wikipedia.org/w/index.php?oldid=445296517 *Contributors*: Alpha Omicron, AugPi, Bryan Derksen, Burn, CBM, CharlesC, Colfulus, Davidhorman, Digichoron, Ferkel, Francis Davey, GregorB, Gwalla, Karlscherer3, Lesnail, MagiMaster, Mercurywoodrose, Mosmas, Mrwojo, Nelson, Nylesheise, Poulpy, Quuxplusone, Seba5618, Short Circuit, Simpsons contributor, Slow Riot, Stepa, Strange but untrue, Verbal, Waldir, Xerxes314, 18 anonymous edits

Elementary cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=453745988 *Contributors*: Cesiumfrog, Dave Feldman, Epsilon0, InverseHypercube, RDBury, Ruud Koot, Salix alba, SebastianHelm, Yugsdrawkcabeht, 19 anonymous edits

Rule 30 *Source*: http://en.wikipedia.org/w/index.php?oldid=452714598 *Contributors*: Branttudor, Captain538, Chris the speller, DataWraith, David Eppstein, Electrified mocha chinchilla, Eouw0o83hf, HairyFotr, Hellbus, Ideogram, InverseHypercube, Joshua Andersen, Lecanard, Lesnail, Marasmusine, Nonenmac, Oliviersc2, P1415926535, Pearle, PiAndWhippedCream, Pleasantville, Querencia, R'n'B, RDBury, Racconman, Rjwilmsi, Rorro, SchuminWeb, Simeon, Skellator, Tabletop, Tarcieri, Tregoweth, 21 anonymous edits

Rule 110 *Source*: http://en.wikipedia.org/w/index.php?oldid=449628794 *Contributors*: !melquiades, Anih, Animum, Avalon, Bob, Bporopat, Bryan Derksen, Concerned cynic, Csl77, DV8 2XL, Dabigkid, David Eppstein, Doceddi, Edonovan, Enisbayramoglu, Epsilon0, Eric119, Ethaniel, Exploto, Genaro.J.Martinez, Gioto, Giraffedata, Glaucus, Hairy Dude, Hobart, Ideogram, Ihope127, Imadeitmyself, Inky, InverseHypercube, Jason Quinn, JohnnyNyquist, Jordo ex, Jrosdahl, Lesnail, Likebox, M a s, Machine Elf 1735, Malcohol, MatthewDBA, Moneky, Oerjan, Pleasantville, Qwertyus, RDBury, Rorro, SeeNoEvil, Simeon, Skellator, Smjg, 51 anonymous edits

Rule 90 *Source*: http://en.wikipedia.org/w/index.php?oldid=447969073 *Contributors*: Calcyman, David Eppstein, Dmcq, Eouw0o83hf, Headbomb, Rjwilmsi, Skellator, Wavelength, 1 anonymous edits

Rule 184 *Source*: http://en.wikipedia.org/w/index.php?oldid=433048911 *Contributors*: Blueshifting, Darguz Parsilvan, David Eppstein, Dzhim, Gen. von Klinkerhoffen, Hzenilc, Jason Recliner, Esq., Kaldari, Karada, Kzollman, Malhonen, Purpy Pupple, Rjwilmsi, Skellator, Steve Quinn, Zaphraud, 11 anonymous edits

Von Neumann cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=402588194 *Contributors*: Abstract Idiot, BeteNoir, Brain, Btyner, CBM, Calcyman, Cryptic, David Eppstein, Ferkel, Geomatician, GhostInTheMachine, Gwalla, Ideogram, Just zis Guy, you know?, Lesnail, Ouro, Pleasantville, Ruud Koot, Shenme, Simpsons contributor, Smjg, Trovatore, William R. Buckley, 13 anonymous edits

Nobili cellular automata *Source*: http://en.wikipedia.org/w/index.php?oldid=413762005 *Contributors*: Calcyman, Mission Fleg, William R. Buckley, 2 anonymous edits

Codd's cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=435875615 *Contributors*: Avono, Calcyman, Charles Matthews, Dirk P Broer, Ferkel, Gaius Cornelius, Heinrichmartin, Kusunose, Leibniz, Lesnail, Quuxplusone, Rjwilmsi, Safemariner, 17 anonymous edits

Langton's loops *Source*: http://en.wikipedia.org/w/index.php?oldid=455478324 *Contributors*: AderakConsteen, Alai, Alpha Omicron, Asymptote, Calcyman, David Eppstein, Ferkel, GregorB, Iminai, Jacob Finn, Jwz, Klosterdev, Mikeblas, Mosmas, Poulpy, Spinningspark, Stepa, Tabletop, Waldir, William R. Buckley, Wingman4l7, Ytrottier, 7 anonymous edits

Von Neumann universal constructor *Source*: http://en.wikipedia.org/w/index.php?oldid=453280537 *Contributors*: Arrenlex, Btyner, Chris the speller, Davemck, David Eppstein, Dr Default, Ferkel, Friginator, Ideogram, Kanzure, Marasmusine, Maverick1701, Pcap, Penwhale, Phantom Hoover, Pleasantville, Robert K S, Ronald King, Serketan, Simpsons contributor, Special-T, StuffOfInterest, Thiseye, Trovatore, Waldir, William R. Buckley, 27 anonymous edits

Firing squad synchronization problem *Source*: http://en.wikipedia.org/w/index.php?oldid=450304258 *Contributors*: David Eppstein, Emurphy42, Giftlite, Hermel, Ideogram, Lecanard, Michael Hardy, Mjancuska, Pleasantville, Quuxplusone, Rjwilmsi, Stdazi, Tr00rle, 20 anonymous edits

Majority problem (cellular automaton) *Source*: http://en.wikipedia.org/w/index.php?oldid=454264298 *Contributors*: David Eppstein, Ezubaric, Gioto, HarryHenryGebel, Kaldari, Michael Hardy, Rjwilmsi, Tamfang, Urmom496, Winchelsea, 2 anonymous edits

Asynchronous cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=436448363 *Contributors*: Clicketyclack, DH85868993, David Eppstein, Dcornforth, False vacuum, Galoubet, Gleishma, Jafet, Jeepday, MrKIA11, Music Sorter, Rjwilmsi, Ykanada, 23 anonymous edits

Automata theory *Source*: http://en.wikipedia.org/w/index.php?oldid=453992431 *Contributors*: A Generic Reality, Aaron Schulz, Ahmad.shahwan, Ahoerstemeier, Alansohn, Allan McInnes, Andrew Eisenberg, Ankog, Arslan asghar, Ashutosh y0078, AxelBoldt, Bci2, Begoon, Bethnim, Bookandcoffee, Bzier, CRGreathouse, ChuckHG, Cominging, Crystallina, Dcoetzee, Deldotvee, DerHexer, Dmcq, Dudesleeper, Déjà Vu, Ehn, Ericszhao1, Eslip17, FoxLogick, Fudo, Gaius Cornelius, Giftlite, GlasGhost, Gmb7-NJITWILL, Gonzonoir, HRV, Helder.wiki, Helix84, Helptry, Hermel, Hjfreyer, Ilyaroz, IvanAndreevich, JHolman, JK the unwise, Jackson, Jagged 85, Jcarroll, Jdoe87, Jeff G., Jeffrey Mall, Jibarraj, Jimbryho, Jitse Niesen, Jjovanw, Johndbritton, Jokes Free4Me, Joseph Solis in Australia, Jpbowen, Jpceayene, Jpvinall, Juniuswikia, KSlayer, KSmrq, Knverma, Konradek, Linas, Ling.Nut, Lobner, MONGO, Magmi, Mangledorf, Maple.writes, Mark lee stillwell, MarkSweep, Marudubshinki, MathMartin, MatthewUND, Mct mht, Mean as custard, Melidee, Mhhwang2002, Michael Devore, Msoos, Musiphil, Nunquam Dormio, Nuttycoconut, Oddity-, Oleg Alexandrov, Omnipaedista, Quoth, Qwertyus, Rjwilmsi, Ruud Koot, Ryanli (usurped), S h i v a (Visnu), SOMNIVM, Saber girl08, Saforrest, Salix alba, Schwarzbichler, Sgkay, Sharaar22, Sietse Snel, Silvonen, Snowolf, Spoon!, Stephen Shaw, Suruena, The Thing That Should Not Be, Thunderboltz, TimBentley, Tobias Bergemann, Toolnut, Ubermonkey, Unbitwise, Valodzka, Vegpuff, Vento, Vojta, Wasbeer, Wjmallard, Yosef Berman, Ze miguel, Zero sharp, Zundark, سعی, 212 anonymous edits

Cyclic cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=446229949 *Contributors*: AmeliaIsland, Bob.lanahan, Borelian, Chadernook, David Eppstein, Ideogram, Keenan Pepper, Kri, Pleasantville, Simpsons contributor, Tamfang, Zakuragi, Zemyla, 3 anonymous edits

Excitable medium *Source*: http://en.wikipedia.org/w/index.php?oldid=432645729 *Contributors*: 4dhayman, Chopchopwhitey, Danwills, Dreadstar, Dysprosia, Firsfron, Jockelinde, Marcus2, Mdd, Ollie holton, Paul August, RuM, Sam Hocevar, SgtPepper967, Vonkje, Wrauscher, 11 anonymous edits

A New Kind of Science *Source*: http://en.wikipedia.org/w/index.php?oldid=450833036 *Contributors*: AaronSw, Adicarlo, AllenDowney, Arbor, Armando-Martin, Astronautics, AxelBoldt, BBiiis08, Baccyak4H, Benanhalt, Bender235, Bevo, Brentt, Bryan Derksen, CRGreathouse, Centrx, Charles Matthews, Chopchopwhitey, Chutzpan, CieloEstrellado, Cimon Avaro, Cln23, Cmdrjameson, Cyan, DJ Clayworth, Daniel11, Dave souza, David Eppstein, David Koller, Dcoetzee, Digitat, Digizen, Diotti, DonSiano, Dudegalea, EagleFan, Ed Poor, Ed g2s, Edonovan, Edward, El Zoof, Emurphy42, Erudecorp, Espen, Eteq, Eyu100, Flammifer, Flatterworld, Flex, Fplay, Fredkinfollower, Genetics411, Geschichte, Glen Worthey, Good Olfactory, GregorB, Grizzly, Guslacerda, Hairy Dude, Hfastedge, Ideogram, Ike, Izno, J.delanoy, JMiall, JimStyle61093475, John Bahrain, Jok2000, JoshuaZ, Jules.lt, Justin W Smith, KJS77, Kku, Kolyma, Kooo, Kovasb, Kraftlos, Labongo, Lightmouse, Lumidek, MakeRocketGoNow, Maxal, Maximus Rex, Mdd, Mike Lin, Mike Schwartz, Mjklin, Mmeri, Nick Dillinger, Ocolon, On you again, Paranoid, Patrickwilken, Pdelong, Pgan002, Pleasantville, Qutezuce, R Lowry, Randomness, Raymondwinn, RedWolf, Reddi, Redgolpe, Revolver, Rich Farmbrough, Richard.decal, Rickjpelleg, Rjwilmsi, Robinh, Roman à clef, Salsa Shark, Sam Hocevar, Simeon, Skoban, Staylor71, Stephan Schulz, Suslindisambiguator, Telso, Tengfred, Tesseran, The Anome, Thue, TittoAssini, Tonesgal, Touisiau, Tregoweth, Trivialist, Tweisbach, Vonkje, Wikiborg, Милан Јелисавчић, 160 anonymous edits

Quantum cellular automata *Source*: http://en.wikipedia.org/w/index.php?oldid=455361156 *Contributors*: Acalamari, Andreas Kaufmann, Bci2, David Eppstein, Favonian, Ferkel, Firsfron, Infinigesimal, Pja35, Rjwilmsi, RobinK, 16 anonymous edits

Coupled map lattice *Source*: http://en.wikipedia.org/w/index.php?oldid=449014095 *Contributors*: Gadget850, Melaen, Michael Hardy, Mrocklin, Nick Number, R'n'B, RHaworth, Torcini, Travdog8, Twri, Zak.estrada, 80 anonymous edits

Movable cellular automaton *Source*: http://en.wikipedia.org/w/index.php?oldid=426139260 *Contributors*: 1ForTheMoney, Asmolin, CRGreathouse, Cirt, Classicalecon, David Eppstein, Firsfron, Laesod, LilHelpa, Martin H., Michael Hardy, RDBury, RHaworth, Rjwilmsi, 10 anonymous edits

Image Sources, Licenses and Contributors

Image:Gospers glider gun.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Gospers_glider_gun.gif *License*: GNU Free Documentation License *Contributors*: Kieff

Image:Torus.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Torus.png *License*: Public Domain *Contributors*: -

Image:CA rule30s.png *Source*: http://en.wikipedia.org/w/index.php?title=File:CA_rule30s.png *License*: GNU Free Documentation License *Contributors*: -

Image:CA rule110s.png *Source*: http://en.wikipedia.org/w/index.php?title=File:CA_rule110s.png *License*: GNU Free Documentation License *Contributors*: -

Image:AC rhombo.png *Source*: http://en.wikipedia.org/w/index.php?title=File:AC_rhombo.png *License*: Creative Commons Attribution 3.0 *Contributors*: Akramm

Image:Oscillator.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Oscillator.gif *License*: GNU Free Documentation License *Contributors*: Original uploader was Grontesca at en.wikipedia

Image:Textile cone.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Textile_cone.JPG *License*: GNU Free Documentation License *Contributors*: -

Image:Brian's brain.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Brian's_brain.gif *License*: Public Domain *Contributors*: Simpsons contributor (talk). Original uploader was Simpsons contributor at en.wikipedia

Image:BriansBrain_oscillator.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:BriansBrain_oscillator.gif *License*: Public Domain *Contributors*: colfulus**?**

File:Gospers glider gun.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Gospers_glider_gun.gif *License*: GNU Free Documentation License *Contributors*: Kieff

File:Game of life block with border.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_block_with_border.svg *License*: Public Domain *Contributors*: None

File:Game of life beehive.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_beehive.svg *License*: Public Domain *Contributors*: None

File:Game of life loaf.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_loaf.svg *License*: Public Domain *Contributors*: None

File:Game of life boat.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_boat.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life blinker.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_blinker.gif *License*: Public domain *Contributors*: JokeySmurf at en.wikipedia

File:game of life toad.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_toad.gif *License*: Public Domain *Contributors*: JokeySmurf

File:game of life beacon.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_beacon.gif *License*: Public domain *Contributors*: JokeySmurf at en.wikipedia

File:game of life pulsar.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_pulsar.gif *License*: Public domain *Contributors*: JokeySmurf at en.wikipedia

File:Game of life animated glider.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_animated_glider.gif *License*: Public Domain *Contributors*: MER-C, RodrigoCamargo, Spikebrennan

File:Game of life animated LWSS.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_animated_LWSS.gif *License*: Public domain *Contributors*: Rodrigo Silveira Camargo a.k.a. RodrigoCamargo at en.wikipedia

File:Game of life fpento.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_fpento.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life diehard.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_diehard.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life acorn.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_acorn.svg *License*: Public Domain *Contributors*: None

File:Game of life glider gun.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_glider_gun.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life infinite1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_infinite1.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life infinite2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_infinite2.svg *License*: Public Domain *Contributors*: Bryan.burgers

File:game of life infinite3.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Game_of_life_infinite3.svg *License*: Public Domain *Contributors*: Original version by w:en:User:Kieff. SVG version by User:FedericoMP.

File:Long gun.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Long_gun.gif *License*: Public Domain *Contributors*: Simpsons contributor (talk)

File:Oscillator.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Oscillator.gif *License*: GNU Free Documentation License *Contributors*: Original uploader was Grontesca at en.wikipedia

File:Golly.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Golly.png *License*: unknown *Contributors*: Original uploader was Simpsons contributor at en.wikipedia

File:LangtonsAnt.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt.png *License*: Public Domain *Contributors*: -

Image:LangtonsAntAnimated.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAntAnimated.gif *License*: Public Domain *Contributors*: Krwawobrody

Image:LangtonsAnt-nColor_RLR_13937.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt-nColor_RLR_13937.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 11:12, 30 September 2009 (UTC)

Image:LangtonsAnt-nColor_LLRR_123157.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt-nColor_LLRR_123157.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 11:11, 30 September 2009 (UTC)

Image:LangtonsAnt-nColor_LRRRRRLLR_70273.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt-nColor_LRRRRRLLR_70273.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 11:13, 30 September 2009 (UTC)

Image:LangtonsAnt-nColor_LLRRRLRLRLLR_36437.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt-nColor_LLRRRLRLRLLR_36437.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 11:13, 30 September 2009 (UTC)

Image:LangtonsAnt-nColor_RRLLLRLLLRRR_32734.png *Source*: http://en.wikipedia.org/w/index.php?title=File:LangtonsAnt-nColor_RRLLLRLLLRRR_32734.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 11:12, 30 September 2009 (UTC)

File:Turmite-111180121010-12536.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-111180121010-12536.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:23, 15 October 2009 (UTC)

File:Turmite-120121010011-8342.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-120121010011-8342.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:28, 15 October 2009 (UTC)

File:Turmite-121021110111-27731.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-121021110111-27731.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:30, 15 October 2009 (UTC)

File:Turmite-121181121020-65932.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-121181121020-65932.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:32, 15 October 2009 (UTC)

File:Turmite-180121020081-223577.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-180121020081-223577.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:33, 15 October 2009 (UTC)

File:Turmite-181181121010-10211.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Turmite-181181121010-10211.png *License*: Creative Commons Zero *Contributors*: Ferkel (talk) 10:35, 15 October 2009 (UTC)

Image:Wireworld two-diodes.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Wireworld_two-diodes.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Thomas Schoch

Image:Animated display.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Animated_display.gif *License*: Public Domain *Contributors*: Simpsons contributor (talk)

Image:Wireworld XOR-gate.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Wireworld_XOR-gate.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Thomas Schoch

Image:Rule0rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule0rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule1rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule1rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule2rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule2rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule3rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule3rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule4rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule4rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule5rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule5rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule6rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule6rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule7rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule7rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule8rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule8rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule9rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule9rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule10rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule10rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule11rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule11rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule12rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule12rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule13rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule13rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule14rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule14rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule15rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule15rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule18rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule18rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule19rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule19rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule22rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule22rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule23rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule23rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule24rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule24rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule25rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule25rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule26rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule26rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule27rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule27rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule28rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule28rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule29rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule29rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule30rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule32rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule32rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule33rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule33rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule34rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule34rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule35rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule35rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule36rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule36rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule37rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule37rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule38rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule38rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule40rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule40rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule41rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule41rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule42rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule42rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule43rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule43rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule44rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule44rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule45rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule45rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule46rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule46rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule50rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule50rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule51rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule51rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule54rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule54rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule56rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule56rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule57rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule57rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule58rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule58rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule60rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule60rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule62rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule62rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule72rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule72rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule73rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule73rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule74rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule74rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule76rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule76rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule77rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule77rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule78rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule78rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule90rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule90rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule94rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule94rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule104rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule104rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule105rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule105rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule106rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule106rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule108rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule108rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule110rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule110rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule122rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule122rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule126rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule126rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule128rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule128rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule130rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule130rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule132rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule132rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule134rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule134rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule136rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule136rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule138rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule138rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule140rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule140rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule142rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule142rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule146rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule146rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule150rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule150rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule152rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule152rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule154rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule154rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule156rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule156rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube
Image:Rule160rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule160rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule162rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule162rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule164rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule164rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule168rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule168rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule170rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule170rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule172rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule172rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule178rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule178rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule184rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule184rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule200rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule200rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule204rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule204rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

Image:Rule232rand.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule232rand.png *License*: Creative Commons Zero *Contributors*: User:InverseHypercube

File:OEISicon light.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:OEISicon_light.svg *License*: Public Domain *Contributors*: Lipedia

File:Textile cone.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Textile_cone.JPG *License*: GNU Free Documentation License *Contributors*: -

Image:Rule 30.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_30.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Simpsons contributor

Image:ca110-structures.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Ca110-structures.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: 4dhayman

Image:ca110-interaction.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Ca110-interaction.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: -

Image:Cts-diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Cts-diagram.jpg *License*: Public Domain *Contributors*: -

Image:R090 rand 0.png *Source*: http://en.wikipedia.org/w/index.php?title=File:R090_rand_0.png *License*: Creative Commons Attribution-Share Alike *Contributors*: Eouw0o83hf

File:Rule 90 gate array.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_90_gate_array.svg *License*: Public Domain *Contributors*: David Eppstein

File:Rule 90 trees.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_90_trees.svg *License*: Public Domain *Contributors*: David Eppstein

Image:R090 pulse wide.png *Source*: http://en.wikipedia.org/w/index.php?title=File:R090_pulse_wide.png *License*: Creative Commons Attribution-Share Alike *Contributors*: eouw0o83hf

File:Highlife replicator.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Highlife_replicator.png *License*: Public domain *Contributors*: Herbee

Image:Rule 184.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_184.png *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

Image:Rule 184 cars.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_184_cars.svg *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

Image:Rule 184 deposition.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_184_deposition.svg *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

Image:Rule 184 annihilation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule_184_annihilation.svg *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

Image:VonNeumann CA demo.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:VonNeumann_CA_demo.gif *License*: Public Domain *Contributors*: Ferkel

Image:VonNeumann CA construction demo.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:VonNeumann_CA_construction_demo.gif *License*: Public Domain *Contributors*: Ferkel (Ferkel (talk))

File:Pixel golly.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Pixel_golly.gif *License*: Public Domain *Contributors*: Simpsons contributor (talk). Original uploader was Simpsons contributor at en.wikipedia

File:Lambda-G.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Lambda-G.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Calcyman (talk)

File:XOR-crossover.png *Source*: http://en.wikipedia.org/w/index.php?title=File:XOR-crossover.png *License*: Public Domain *Contributors*: Calcyman (talk). Original uploader was Calcyman at en.wikipedia

Image:Codd CA RepeaterEmitter.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Codd_CA_RepeaterEmitter.gif *License*: Public Domain *Contributors*: Ferkel (Ferkel (talk))

Image:Codd CA ConstructionArm.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Codd_CA_ConstructionArm.gif *License*: GNU General Public License *Contributors*: Ferkel (talk) 00:16, 4 February 2009 (UTC)

Image:Langtons Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Langtons_Loop.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:13, 4 February 2009 (UTC)

Image:Langtons Loop Colony.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Langtons_Loop_Colony.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:16, 4 February 2009 (UTC)

Image:Byl Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Byl_Loop.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:20, 4 February 2009 (UTC)

Image:Chou-Reggia Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Chou-Reggia_Loop.png *License*: Public Domain *Contributors*: Ferkel (talk) 20:21, 4 February 2009 (UTC)

Image:Tempesti Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Tempesti_Loop.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:24, 4 February 2009 (UTC)

Image:Perrier Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Perrier_Loop.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:25, 4 February 2009 (UTC)

Image:SDSR Loop.png *Source*: http://en.wikipedia.org/w/index.php?title=File:SDSR_Loop.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:26, 4 February 2009 (UTC)

Image:Evoloop closeup.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Evoloop_closeup.png *License*: GNU General Public License *Contributors*: Ferkel (talk) 20:28, 4 February 2009 (UTC)

Image:Nobili Pesavento 2reps.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Nobili_Pesavento_2reps.png *License*: Public Domain *Contributors*: Ferkel

Image:Pesavento replicator inherited mutations.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Pesavento_replicator_inherited_mutations.png *License*: Public Domain *Contributors*: Ferkel

Image:FiringSquadProblem.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:FiringSquadProblem.gif *License*: Public Domain *Contributors*: -

Image:FSSP 1d1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:FSSP_1d1.svg *License*: GNU Free Documentation License *Contributors*: -

Image:Rule30 sync.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_sync.png *License*: Public Domain *Contributors*: David Cornforth

Image:Rule30 RAI.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_RAI.png *License*: Public Domain *Contributors*: David Cornforth

Image:Rule30 RAO.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_RAO.png *License*: Public Domain *Contributors*: David Cornforth

Image:Rule30 cyclic.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_cyclic.png *License*: Public Domain *Contributors*: David Cornforth

Image:Rule30 clock.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_clock.png *License*: Public Domain *Contributors*: David Cornforth

Image:Rule30 self.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Rule30_self.png *License*: Public Domain *Contributors*: David Cornforth

Image:DFAexample.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DFAexample.svg *License*: Public Domain *Contributors*: Cepheus

Image:1dCCA-n4.png *Source*: http://en.wikipedia.org/w/index.php?title=File:1dCCA-n4.png *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

Image:Super cyclic.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Super_cyclic.gif *License*: Public Domain *Contributors*: Simpsons contributor (talk)

Image:2dCCA-n16.png *Source*: http://en.wikipedia.org/w/index.php?title=File:2dCCA-n16.png *License*: Public Domain *Contributors*: Original uploader was David Eppstein at en.wikipedia

File:ExcitableCellularAutomaton.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:ExcitableCellularAutomaton.gif *License*: Creative Commons Attribution 3.0 *Contributors*: 4dhayman

Image:Cml2e.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Cml2e.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: -

Image:Cml3a.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Cml3a.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: -

Image:Frozenchaos logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Frozenchaos_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:PatternSelection logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:PatternSelection_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:BrownMotionDefect logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:BrownMotionDefect_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:DefectTurbulence logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:DefectTurbulence_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:Spatiotemporal Intermittency logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Spatiotemporal_Intermittency_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:Spatiotemporal Intermittency logmap2.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:Spatiotemporal_Intermittency_logmap2.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:SpatiotemporalChaos fullydevd logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:SpatiotemporalChaos_fullydevd_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: -

Image:TravelingWave logmap.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:TravelingWave_logmap.JPG *License*: Creative Commons Attribution 3.0 *Contributors*: Travdog8

File:MCA friction net.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_friction_net.gif *License*: GNU Free Documentation License *Contributors*: Laesod

File:MCA elements.png *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_elements.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA neighbors.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_neighbors.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA sh1.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_sh1.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA sh2.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_sh2.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA switch.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_switch.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA neighbour in pair.png *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_neighbour_in_pair.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA Deformation in Pair of Automata.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_Deformation_in_Pair_of_Automata.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA Irreversible Deformation.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_Irreversible_Deformation.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA response function of automata.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_response_function_of_automata.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA cyclic schem.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_cyclic_schem.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

File:MCA cyclic diag.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:MCA_cyclic_diag.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Laesod

License