Session 7

Cellular Automata

Juan Li
Communications Laboratory, HUT

First beginning with introduction to the history of cellular automata, this paper pesents several important ideas in this field. Then a brief analysis of 1-D cellular automata is given. Finally, some more advanced applications of cellular automata are demonstrated.

7.1   Introduction

The increasing prominence of computers has led to a new way of looking at the world. This view sees nature as a form of computation. That is, we treat objects as simple computers, each obeying its own set of laws.

A computer follows rules. At each moment, the rules determine exactly what the computer will do next. We say that a computer is an example of an automaton. Other, simpler examples of automata also exist[1]. These more abstract rule-following devices can be easier to study using computers, and they can be interesting to study in their own right. One type of automaton that has received a lot of attention is *cellular automata.* For one thing, they

---

[1]Automata is the plural of automaton. While the word "automaton" conjures up the image of a mechanical toy or a soulless organism, in computer science it has a very precise meaning. It refers to all machines whose output behaviour is not a direct consequence of the current input, but of some past history of its inputs. They are characterised as having an internal state which is a repository of this past experience. The inner state of an automaton is private to the automaton, and is not available to an external observer.

make pretty pictures. For another, they are related to exciting new ideas such as artificial life and the edge of chaos. For a fairly simple example see [1].

The "cellular automaton" provides a way of viewing whole populations of interacting "cells", each of which is itself a computer (automaton). By building appropriate rules into a cellular automaton, we can simulate many kinds of complex behaviours, ranging from the motion of fluids governed by the Navier-Stokes equations to outbreaks of starfish on a coral reef.

7.2   History of Cellular Automata

7.2.1   Spectacular historical automata

The earliest automata were mechanical devices that seemed to demonstrate lifelike behaviour. They took advantage not only of gears, but also of gravity, hydraulics, pulleys and sunlight — the effect could be dazzling, as with the extraordinary clock of Berne: Created in 1530, this massive timepiece hourly disgorged a dazzling pageantry of automata figures, beginning with a crowing cock and followed by a procession in which the nodding head of a clock king allowed the passage of a parade of spear-wielding bear cubs and a ferocious lion!

The most famous of early automata was the creation of Jacques de Vaucanson, who in 1738 dazzled Paris with "an artificial duck made of gilded copper who drinks, eats, quacks, splashes about the water, and digests his food like a living duck." The complexity of this duck was enormous — there were over four hundred moving pieces in a single wing.

7.2.2   Early history of cellular automata

Mathematician Stanislaw M. Ulam liked to invent pattern games for the computer at Los Alamos. Given certain fixed rules, the computer would print out ever-changing patterns. Many patterns grew almost as if they were alive. A simple square would evolve into a delicate, coral-like growth. Two patterns would "fight" over territory, sometimes leading to mutual annihilation. He developed 3-D games too, constructing thickets of coloured cubes as prescribed by computer. He called the patterns "recursively defined geometric objects". Ulam's games were cellular games. Each pattern was composed of square (or triangular or hexagonal) cells. In effect, the games were played on limitless

chessboards. All growth and change of patterns took place in discrete jumps. From moment to moment, the fate of a given cell depended only on the states of its neighbouring cells. The advantage of the cellular structure is that it allows a much simpler physics. Without the cellular structure, there would be infinitely many possible connections between components.

Ulam suggested that John von Neumann "construct" an abstract universe for his analysis of machine reproduction. It would be an imaginary world with self-consistent rules, as in Ulam's computer games. It would be a world complex enough to embrace all the essentials of machine operation, but otherwise as simple as possible. Von Neumann adopted an infinite chessboard as his universe. Each square cell could be in any of a number of states corresponding roughly to machine components. A "machine" was a pattern of such cells. The rules governing the world would be a simplified physics. A proof of machine reproduction would be easier to devise in such an imaginary world, as all the nonessential points of engineering would be stripped away [1]. Ironically, the name von Neumann is now strongly associated with the old-fashioned, single-CPU computer architecture. He was also the major pioneer in parallel computing via his research on arrays of computer or cellular automata.

In 1944, von Neumann was introduced to electronic computing via a description of the ENIAC by Goldstine. Shortly, he formed a group of scientists headed by himself, to work on problems in computers, communications, control, time-series analysis, nd the "communications and control aspects of the nervous system". The last topic was included due to his great interest in the work on neural networks of McCulloch and Pitts. In 1946, von Neumann proceeded to design the EDVAC (Electronic Discrete Variable Computer) which was the first design of the ideas on automata developed by Post (1936) and Turing (1936), he had commenced studies on the complexity required for a device or system to be self-reproductive. These studies also included work on the problem of organizing a system from basically unreliable parts (a field of study which we now know as "fault tolerant computing"). At first, von Neumann investigated a continuous model of a self-reproducing automaton based on a "system of non-linear partial differential equations, essentially of the diffusion type". He also pursued the idea of a kinematic automaton which could, using a description of itself, proceed to mechanically assemble a duplicate from available piece parts.

When von Neumann found it difficult to provide the rigorous and explicit rules and instructions needed to realize such an automaton and when it became evident that the value of such an automata would be moot, he redirected

his efforts towards a model of self-reproduction using an array of computing elements. Both Burks and Goldstine confirm that the idea of such an array was suggested to von Neumann by Stanislaw Ulam. Von Neumann was also attracted to this idea of using parallelism because he saw that it would eventually lead to high computational speed. By 1952 he had put his ideas in writing and in 1953 described them more fully in his lectures at Princeton University. Unfortunately, his premature death in 1957 prevented him from completely achieving his goals. Thus, it can be said that, in the early 1950s, von Neumann conceived of the cellular automata [2].

7.2.3   Von Neumann's self-reproducing cellular automata

The central problem of self-reproduction is the problem of *infinite regress.* Living organisms are finite and reproduce in a finite time. Any machine, whether in the real world, von Neumann's cellular space, or Life's cellular space, is likewise finite. If self-reproduction involves infinities, then it is pointless to look for self- reproducing machines.

The problem with machine reproduction is that the universal constructor is a mindless robot. It has to be told very explicitly what to do. If such a machine were given a description of an alleged self-reproducing machine, this constructor needs to understand that it is supposed to be reproducing the machine, including its description. Von Neumann's solution was to append a "supervisory unit" to the machine to handle precisely such tasks. Infinities are avoided because the machine description does not try to encapsulate itself, but is interpreted in two ways. It is first interpreted literally, as a set of directions to be followed in order to make a certain type of machine. Once the self-reproduction has entered the second phase, the instructions are ignored, and the code is treated merely as data for the copying process.

Von Neumann began with a horizonless grid, with each cell in an inactive state. An organism was then introduced, covering two hundred thousand cells. The details of this creature were represented by different states of individual cells — there were 29 possible states. The different combinations of these states governed the behaviour of the organism, and defined the organism itself.

It was shaped like a box with a very long tail; the box, about eighty cells long by four hundred cells wide, contained suborganisms:

- A factory, gathering "materials" from the environment and arranging

them according to instructions from another suborganism,

- A duplicator, reading informational instructions and copying them, and

- A computer, the control apparatus.

These took up only a quarter of the creature's total number of cells. The rest of the cells were in a single-file line of 150,000 cells, acting as a code for the instructions to construct the entire organism.

Once this automaton was embedded in the grid, each cell, as an individual finite state machine, began to follow the rule that applied to it. The effect of these local behaviours caused a global behaviour to emerge: The self-reproducing structure interacted with neighbouring cells and changed some of their states. It transformed them into the materials — in terms of cell states — that made up the original organism. The tail of the cell contained instructions for the body of the creature. Eventually, by following rules of transition (drawn up by Von Neumann), the organism made a duplicate of its main body; information was passed through an "umbilical cord", from parent to child. The last step in the process was the duplication of the tail, and the detachment of the "umbilical cord". Two identical creatures, both capable of self-reproduction, were now on the grid.

7.2.4   Conway's Game of Life

Life was devised in 1970 by John Horton Conway, a young mathematician at Gonville and Caius College in Cambridge, and it was introduced to the world via two of Martin Gardner's columns in Scientific American (October 1970 and February 1971). Although it was true that von Neumann's automaton qualified as a universal computer (it could emulate any describable function of any other machine by the use of a set of logical rules), the organism itself was very complex, with its two hundred thousand cells in any of twenty-nine states. Conway suspected that a cellular automaton with universal computing capabilities might be simpler. The key to this simplicity would be the rules that dictated survival, birth and death.

The game Life is a simple 2-D analogue of basic processes in living systems. The game consists in tracing changes through time in the patterns formed by sets of "living" cells arranged in a 2-dimensional grid. Any cell in the grid may be in either of two states: "alive" or "dead". The state of each cell changes from one generation to the next depending on the state of its

immediate neighbours. The rules governing these changes are designed to mimic population change.

The behaviour in Life is typical of the way in which many cellular automata reproduce features of living systems. That is, regularities in the model tends to produce order: Starting from an arbitrary initial configuration, order (i.e., patches of pattern) usually emerges fairly quickly. Ultimately, most configurations either disappear entirely or break up into isolated patterns that are either static or else cycle between several different forms with a fixed period.

Conway tried many different numerical thresholds for birth and survival. He had three objectives:

- First, Conway wanted to make sure that no simple pattern would obviously grow without limit. It should not be easy to prove that any simple pattern grows forever.

- Second, he wanted to ensure, nonetheless, that some simple patterns do grow wildly. There should be patterns that look like they might grow forever.

- Third, there should be simple patterns that evolve for a long time before stabilising. A pattern stabilised by either vanishing completely or producing a constellation of stable objects.

If Life's rules said that any cell with a live neighbour qualifies for a birth and no cell ever dies, then any initial pattern would grow like a crystal endlessly. If on the other hand the rules were too anti-growth, then everything would die out — Conway contrived to balance the tendencies for growth and death. It turned out that these objectives were fulfilled when the following rules were applied:

STASIS If, for a given cell, the number of on neighbours is exactly two, the cell maintains its status into the next generation. If the cell is on, it stays on, if it is off, it stays off.

GROWTH If the number of on neighbours is exactly three, the cell will be on in the next generation. This is regardless of the cell's current state.

DEATH If the number of on neighbours is 0, 1, or 4–8, the cell will be off in the next generation.

In the real world, matter-energy cannot be created or destroyed. If a colony of bacteria grow to cover a Petri dish, it is only by consuming nutrient. The

Petri dish as a whole weighs the same before and after. However, no such restriction applies in Life, the amount of "matter" in the Life universe can fluctuate arbitrarily.

Simple rules can have complex consequences. The simple set of rules in Life has such a wealth of implications that it is worth examining in detail. Life is *forward-deterministic:* This means that a given pattern leads into one, and only one sequel pattern. The Game of Life is not backward-deterministic; a pattern usually has many patterns that may have preceded it. In short, a configuration has only one future, but usually many possible pasts. This fact is responsible for one of the occasional frustrations of playing Life. Sometimes you will see something interesting happen, stop the program, and be unable to backtrack and repeat it. There is no simple way you can program a computer to go backward from a Life state — there are too many possibilities.

Conway had tried to design Life's rules so that unlimited growth patterns were possible. It was not immediately obvious that he had succeeded, though. Conway conceived of two ways of demonstrating that unlimited growth is possible, if indeed it is. Both ways involved finding a pattern that does grow forever, yet in a disciplined, predictable manner.

In 1970, in response to Conway's challenge in the Scientific American that a finite initial configuration of Life would not be able to generate infinite populations, R.W. Gosper used a DEC PDP-6 computer to run Life simulations far quicker than could be done by hand. Eventually, they found the "glider gun", which generated gliders continually, and a "puffer train" that moved on the grid, leaving behind a constant trail of live cells. They even found "puffer trains that emitted gliders that collided to make glider guns which then emitted gliders, but in a quadratically increasing number ...".

From this breakthrough, Conway could prove that Life could indeed support *universal computation.* Using glider streams to represent bits, he was able to produce the equivalent of and-gates, or-gates and not-gates, as well as an analogue of a computer's internal storage.

7.2.5 Stephen Wolfram and 1-dimensional cellular automata

Stephen Wolfram worked with a one-dimensional variant of von Neumann's cellular automata; this was fully horizontal and occurred on a single file line. Each cell touched only two other cells, its two immediate neighbours on either side, and each succeeding generation was represented by the line

underneath the preceeding one. A cell in generation 2 would determine its state by looking at the cell directly above it, ie., in generation 1, and that cell's two neighbours. Thus, there are eight possible combinations of the states of those 3 cells, ranging from "000" (all off) to "111" (all on).

Since there are 8 possible states for the ancestors of any given cell, and these states may result in one of two states (1 or 0), there are 256 possible rulesets for this type of cellular automata. Wolfram explored them all. Some of those rules quickly resolved themselves into boring configurations, for example, all dead cells or all live cells. Wolfram called these Class 1 cellular automata. Another variation was some other frozen configuration where initial activity ceased and stable structures reigned — Wolfram designated these as Class 2. However, other configurations broke up into relatively disordered patterns, resembling video noise, but sometimes scattered with inverted triangles. This was Class 3.

There was a final class, Class 4, of cellular automata that displayed behaviour that was not disordered, but complex, and sometimes long-lived. These were capable of propagating information, and included all cellular automata that supported universal computers (see Fig. 7.1).

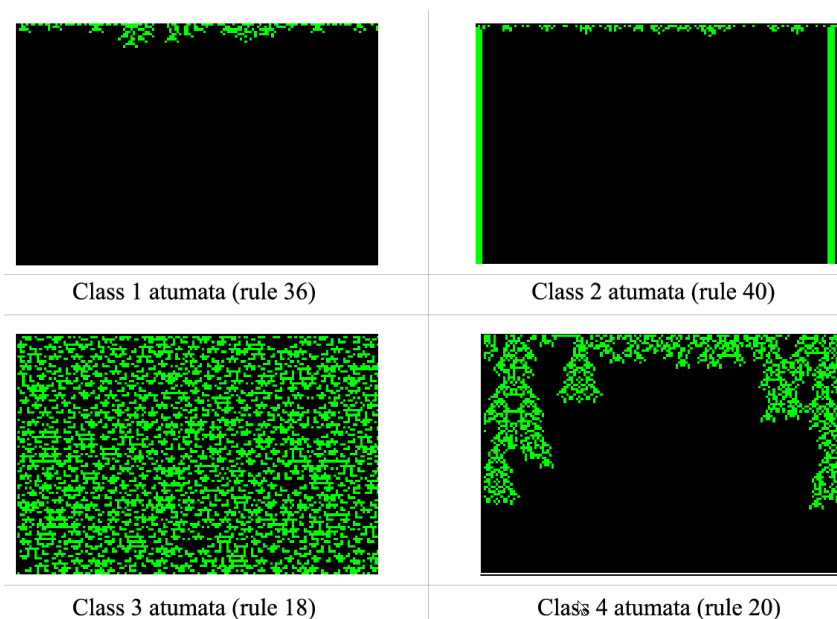| Class 1 atumata (rule 36) | Class 2 atumata (rule 40) |
| Class 3 atumata (rule 18) | Class 4 atumata (rule 20) |

Figure 7.1: Wolfram's four classes of cellular automata

7.2.6 Norman Packard's Snowflakes

Norman Packard was a physicist working with Stephen Wolfram at the Institute for Advanced Study, and chose to apply information theory to cellular automata to emulate snowflakes. He reasoned that firstly, it was not the case that every snowflake was identical. However, if every snowflake had a random structure, the information from each snowflake would be meaningless. Therefore, there is a syntax within snowflakes, several main types of structures, which are capable of containing individual variations.

Packard discovered that different weather conditions result in snowflakes taking on different general aspects. One set of conditions yields configurations that look like plates, another determines snowflakes shaped like collections of rods, and another yields dendritic stars. He wrote a cellular automaton simulation in which the "off" cells, those with a value of "0", represented water vapour, and the "on" cells, those assigned a value of "1", represented ice, and appeared on the screen in colour. The snowflake would grow outwards from its boundary. A typical set of rules initiated a scan of a cell's neighbourhood, summed the values of the surrounding cells, and filled the new cells with either ice or water vapour, depending on whether the toal was odd or even.

The resulting artificial snowflakes lacked the complexity of real snowflakes, particularly those with structures based on patterns of needle-like shapes — but they did have plates and dendrites growing from the corners of the plates, from which more dendrites grew; they were easily idenfiable as snowflakes.

7.3 Cellular automata in technical terms

A cellular automaton is an array of identically programmed automata, or "cells", which interact with one another. The arrays usually form either a 1-dimensional string of cells, a 2-D grid, or a 3-D solid. Most often the cells are arranged as a simple rectangular grid, but other arrangements, such as a honeycomb, are sometimes used.

The essential features of a cellular automaton ("CA" for short) are:

- Its STATE is a variable that takes a different value for each cell. The state can be either a number or a property. For instance if each cell represents part of a landscape, then the state might represent (say) the number of animals at each location or the type of forest cover growing there.

- Its NEIGHBOURHOOD is the set of cells that it interacts with. In a grid these are normally the cells physically closest to the cell in question. Below, some simple neighbourhoods (cells marked n) of a cell (C) in a 2-D grid are shown:

```
      n               n   n   n          n   n   n   n
  n   C   n           n   C   n          n   C   n   n   n   n
      n               n   n   n          n   n   n   n
```

- Its PROGRAM is the set of rules that defines how its state changes in response to its current state, and that of its neighbours.

Example — Fabric patterns

Imagine a CA consisting of a line of cells as follows:

- States: 0 or 1

- Neighbourhood: The two adjacent cells (or "n C n")

- Rules: The following list shows the new state of a cell for each possible local "configuration", i.e. arrangement of states for the cell and its two neighbours. Because there are two possible states (0 or 1) for each of 3 cells, there are $2^3 = 8$ rules required:

```
0  0  0  →  0        0  0  1  →  1        0  1  0  →  1
0  1  1  →  0        1  0  0  →  1        1  0  1  →  1
1  1  0  →  0        1  1  1  →  0
```

Suppose that we start with just a single cell in state 1. Then here is how the array will change with time (see Fig. 7.2; here "." denotes 0 for clarity).

When we plot the successive states as shown above, some properties emerge:

Self-organization: Notice that when we plot the successive states as shown in the above example a pattern emerges. Even if the line of cells starts with a random arrangement of states, the rules force patterns to emerge; the pattern depends on the set of rules, for examples see Fig. 7.3.

```
Time 0 :   . . . . . . . . . . . 1 . . . . . . . . . . .
Time 1 :   . . . . . . . . . . 1 1 1 . . . . . . . . . .
Time 2 :   . . . . . . . . . 1 . . . 1 . . . . . . . . .
Time 3 :   . . . . . . . 1 1 1 . 1 1 1 . . . . . . . .
Time 4 :   . . . . . . 1 . . . 1 . . . 1 . . . . . . .
Time 5 :   . . . . . 1 1 1 . 1 1 1 . 1 1 1 . . . . . .
Time 6 :   . . . . 1 . . . 1 . . . 1 . . . 1 . . . .
```
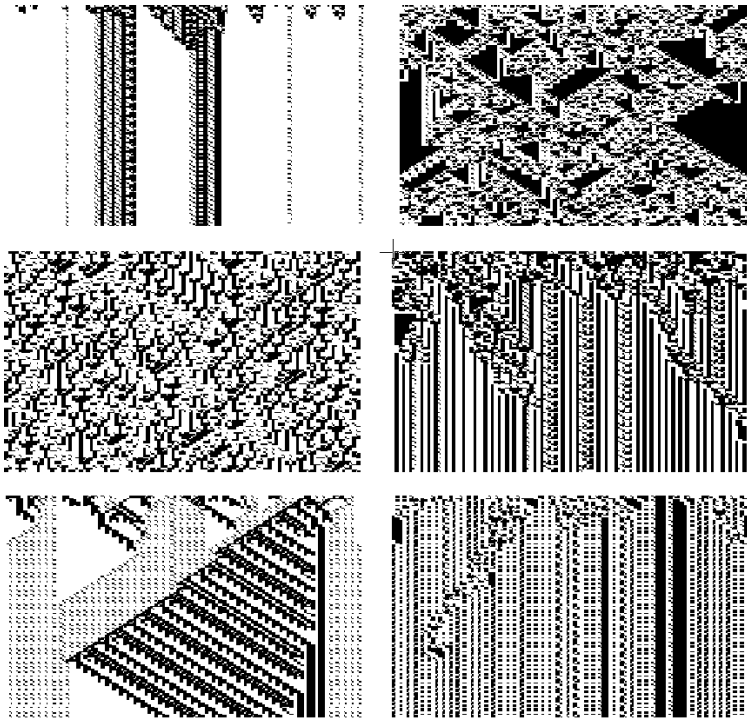
Figure 7.2: Evolution in a one-dimensional CA

Figure 7.3: Fabric patterns

Life-like behaviour: Empirical studies by Wolfram and others show that even the simple linear automata behave in ways reminiscent of complex biological systems. For example, the fate of any initial configuration of a cellular automaton is either

1. to die out;

2. to become stable or cyclic with fixed period;

3. to grow indefinitely at a fixed speed;

4. to grow and contract irregularly.

"Thermal behavior": In general, models that force a change of state for few configurations tend to "freeze" into fixed patterns, whereas models

that change the cell's state in most configurations tend to behave in a more active "gaseous" way. That is, fixed patterns do not emerge.

In other words, a cellular automaton is a discrete dynamical system. Space, time, and the states of the system are discrete. Each point in a regular spatial lattice, called a cell, can have any one of a finite number of states. The states of the cells in the lattice are updated according to a local rule. That is, the state of a cell at a given time depends only on its own state one time step previously, and the states of its nearby neighbors at the previous time step. All cells on the lattice are updated synchronously. Thus the state of the entire lattice advances in discrete time steps [4].

7.4   A Mathematical analysis of a simple cellular automaton

As an example of a cellular automaton, consider a line of sites, each with value 0 or 1 (Fig. 7.4). Take the value of a site at position i on time step t to be a^i. One very simple rule for the time evolution of these site values is

$$a_i^{(t+1)} = a_{i-1}^{(t)} + a_{i+1}^{(t)} \mod 2, \tag{7.1}$$

where mod 2 indicates that the 0 or 1 remainder after division by 2 is taken. According to this rule, the value of a particular site is given by the sum modulo 2 (or, equivalently, the Boolean algebra "exclusive or") of the values of its left and right hand nearest neighbor sites on the previous time step. The rule is implemented simultaneously at each site. Even with this very simple rule quite complicated behavior is nevertheless found.
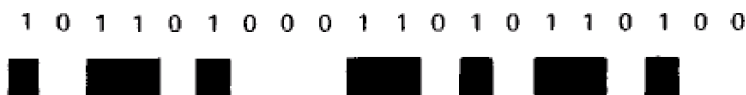

Figure 7.4: A typical configuration in the simple cellular automaton described by (7.1)

The cellular automaton rule of (7.1) is particularly simple and allows a rather complete mathematical analysis. The analysis proceeds by writing for each configuration a characteristic polynomial

$$A(x) = \sum_{i=0}^{N-1} a_i x^i \tag{7.2}$$

where x is a dummy variable, and the coefficient of x^i is the value of the site at position i. In terms of characteristic polynomials, the cellular automaton rule (7.1) takes on the particularly simple form

$$A^{(t+1)}(x) = T(x)A^{(t)}(x) \qquad \text{mod} \qquad (x^N - 1) \tag{7.3}$$

where

$$T(x) = (x + x^{-1}), \tag{7.4}$$

and all arithmetic on the polynomial coefficients is performed modulo 2. The reduction modulo $x^N - 1$ implements periodic boundary conditions. The structure of the state transition diagram may then be deduced from algebraic properties of the polynomial $T(x)$. Since a finite cellular automaton evolves deterministically with a finite total number of possible states, it must ultimately enter a cycle in which it visits a sequence of states repeatedly. Such cycles are manifest as closed loops in the state transition graph. The algebraic analysis of Martin *et al.* shows that for the cellular automaton of (7.1) the maximal cycle length (of which all other cycle lengths are divisors) is given for odd N

$$2^{\text{sord}_{N(2)}} - 1. \tag{7.5}$$

Here $\text{sord}_{N(2)}$ is a number theoretical function defined to be the minimum positive integer j for which $2^j = \pm 1$ modulo N. The maximum value of $\text{sord}_{N(2)}$, typically achieved when N is prime, is $\frac{(N-1)}{2}$. The maximal cycle length is thus of order $2^{\frac{N}{2}}$, approximately the square root of the total number of possible states 2^N.

An unusual feature of this analysis is the appearance of number theoretical concepts. Number theory is inundated with complex results based on very simple premises. It may be part of the mathematical mechanism by which natural systems of simple construction yield complex behavior.

7.5 Applications

Cellular automata applications are diverse and numerous. The laws in our Universe are only partly known and appear to be highly complex, whereas in a cellular automaton laws are simple and completely known. One can then test and analyse the global behaviour of a simplified universe, for example:

- Simulation of gas behaviour. A gas is composed of a set of molecules whose behaviour depends on the one of neighbouring molecules.

- Study of ferromagnetism according to Ising model. This model (1925) represents the material as a network in which each node is in a given magnetic state. This state — in this case one of the two orientations of the spins of certain electrons — depends on the state of the neighbouring nodes.

- Simulation of percolation process.

- Simulation of forest fire propagation.

- Conception of massive parallel computers.

- Simulation and study of urban development.

- Simulation of crystallisation process.

In different fields, cellular automata can be used as an alternative to differential equations. Cellular automata can also be used as graphic generators[2]. The several images in Fig. 7.5 show some graphic effects.
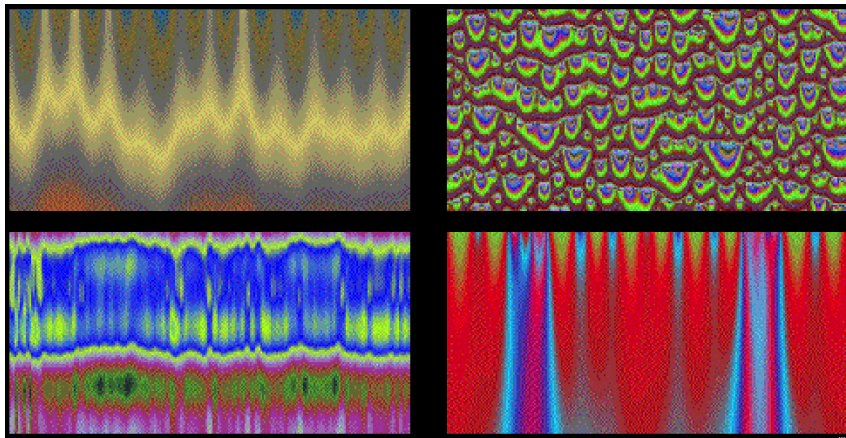

Figure 7.5: Some pictures generated by cellular automata

---

[2]According to Rucker et Walker *op. cit., "In five years you won't be able to watch television for an hour without seeing some kind of cellular automata".*

Bibliography

1. http://math.hws.edu/xJava/CA/

2. Kendall Preston, Jr. and Michael J.B.Duff, *Modern Cellular Automata: Theory and Applications*. Plenum Press, New York, 1984.

3. http://life.csu.edu.au/complex/tutorials/tutorial1.html

4. http://shakti.trincoll.edu/ pkenned3/alife.html

5. http://math.hws.edu/xJava/CA/CA.html

6. http://www.tu-bs.de/institute/WiR/weimar/ZAscriptnew/intro.html

7. http://www.stephenwolfram.com/publications/articles/ca/85-two/2/text.html

8. http://www.brunel.ac.uk/depts/AI/alife/al-ca.htm

9. http://www.rennard.org/alife/english/acintrogb01.html

10. http://www.ce.unipr.it/pardis/CNN/cnn.html

11. http://cafaq.com/soft/index.shtml

12. N. Howard, R. Taylor, and N. Allinson.*The design and implementation of a massively-parallel fuzzy architecture*. Proc. IEEE, pages 545-552, March 1992