# DT8114 - PhD Seminar in Computer Science

STEFANO NICHELE
*Department of Computer and Information Science*
*Norwegian University of Science and Technology*
*Sem Sælandsvei 7-9, NO-7491*
*nichele@idi.ntnu.no*

**Abstract:** *this document represents the final essay for the PhD course DT8114 – PhD Seminar in Computer Science, which is part of my doctoral study program. This report aims to give an overview of the main topics in the course syllabus, which is composed by the following papers and books:*

- *Langton C. - Artificial Life, pag. 1-48 - Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87), Los Alamos, NM, USA, September 1987. Santa Fe Institute Studies in the Sciences of Complexity 6 Addison-Wesley 1989, ISBN 0-201-09346-4*

- *Lindenmayer A. - Developmental Models of Multicellular Organisms: A Computer Graphics Perspective, pag. 221-250 - Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87), Los Alamos, NM, USA, September 1987. Santa Fe Institute Studies in the Sciences of Complexity 6 Addison-Wesley 1989, ISBN 0-201-09346-4*

- *Langton C. - Computation at the Edge of Chaos: phase transition and emergent computation, pag. 12-37 - Physica D 42 1990 — Elsevier Science Publisher B.V. (North-Holland)*

- *Kumar S. and Bentley P. - On Growth, Form and Computers - Elsevier Academic Press, Year 2003 – 444 pages – ISBN 0-12-428765-4*

- *Mainzer K. - Thinking in Complexity – Springer, Year 1994 – 482 pages – ISBN 978-3-540-72227-4*

# Table of Content



**DISCLAIMER**

Some figures in this report are taken from the books and papers in the course syllabus.

# Introduction on Artificial Life

## *What is Artificial Life?*

"Life made by Man rather than by Nature". This is the naive definition of Artificial Life given by Langton in [1]. Nowadays, the biologists study the principles of life based on carbon-chain chemistry, which is actually the only kind of life available on Earth. It is almost impossible to find out general principles if there is only a single example available to study. Another way of creating a "different" kind of life is to simulate or recreate the biological phenomena using computers, trying to reproduce the behaviour of living systems. From this point of view, the domain of study switches from life-as-we-know-it to life-as-it-could-be.

In biology, a living organism is considered as a complex biochemical machine composed by organs, and then tissues, cells, until molecules. This is a top down approach, where the analytical process moves downward from the macro to the micro-world. In Artificial Life the approach is different. The key concept is to put together simple elements governed by local rules and consider the organism as the collection of this large population of small machines that interact one another eventually creating a complex global behaviour, a life-like behaviour. In artificial life machines there is no central controller, the global behaviour emerges out of the interactions among a great number of simple elements.

## *Key Principles*

The main features of an artificial life model are:

- The population is composed by a collection of simple elements;
- There is no central controller that directs all the elements in the population;
- Each element only knows the status of few neighbours and it reacts depending on local situations;
- There are no rules for the global behaviour, only local rules;
- The global behaviour is emergent from the local behaviours.

Since artificial life aims to generate a life-like behaviour, it is not necessary to "bring" life to a machine; rather it is important to organize a population of machines in a way that their interactions and their global behaviour seem alive.

## *History of Artificial Life*

It is in mankind's psyche to try to simulate the nature's behaviour using artefacts or mechanical technologies. Egyptians have built tools able of autonomous behaviour for measuring time, using water clocks called Clepsydra. After, Ctesibius of Alexandria developed a water powered clock using some hydraulic technology. Clocks have been considered the most complicated device and the most advanced application of technology in every era.

It is in 1735 that Jacques de Vaucanson tried to reproduce life artificially, creating and artificial duck composed of complex mechanical mechanisms. In Figure 1 two representations of Vaucanson's duck are shown. Unfortunately, his work was lost and in 1847 Reichsteiner attempted to build a new mechanical duck capable of simulating the behaviour of a real duck.

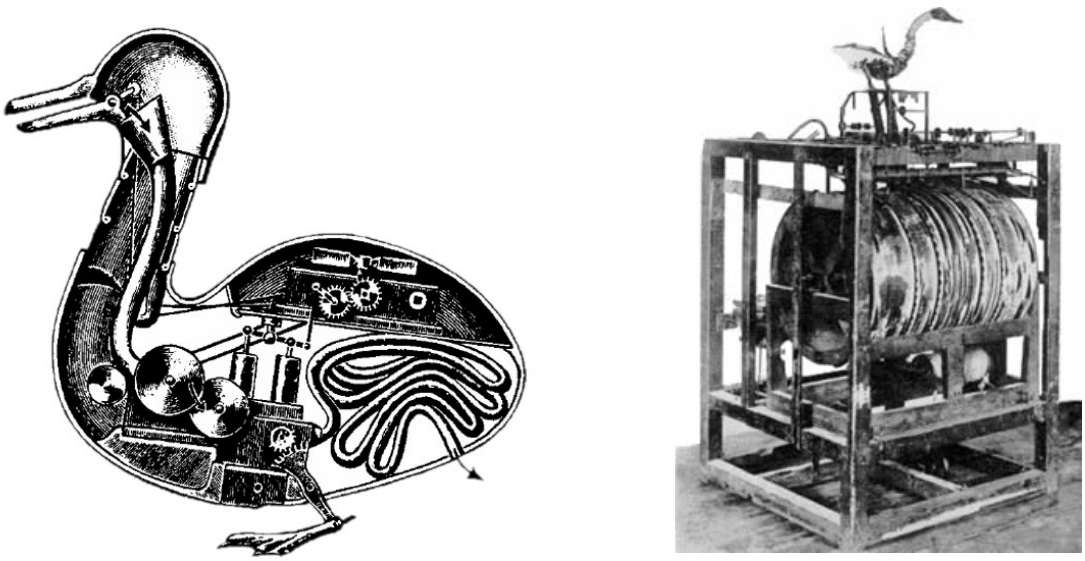


*FIGURE 1: Vaucanson's artificial duck [1].*

It turned out that one of the most important tasks was to control and to sequence the actions of the automata. Control mechanisms in the beginning were really simple but eventually they became programmable.

In the 20$^{th}$ century, Turing, Kleene, Church, Godel and others applied logic to abstract the control mechanisms, formalizing the notion of procedure (or program). Nowadays, despite the physical representation and the actual construction material of the machine, the logical equivalent is an algorithm. In a modern computer, a general purpose machine, the computer itself cannot behave in a specific way without being programmed. In other words, computers are able to generate different behaviours and implement different abstract machines, depending on the specifications represented by

the program. Of course not all behaviours are computable; there are some which cannot be formally specified in order to obtain a machine that will exhibit it. This most famous example of this principle is the halting problem specified by Turing [6].

In 1943, Mc.Culloch and Pitts [7], under Turing's influence, presented a new model of neural networks where every single artificial neuron was a threshold unit with excitatory and inhibitory synapses. As represented in Figure 2, a neuron fires an impulse *y* along its axon at time *n+1* if the weighted sum of its inputs *x1,...,xm* and weights *w1,...,wm* at time *n* exceeds the threshold θ. Those abstractions turned out to be very useful to accomplish complex tasks (complex for computers) such as pattern recognition or other perception tasks performed by the human brain.

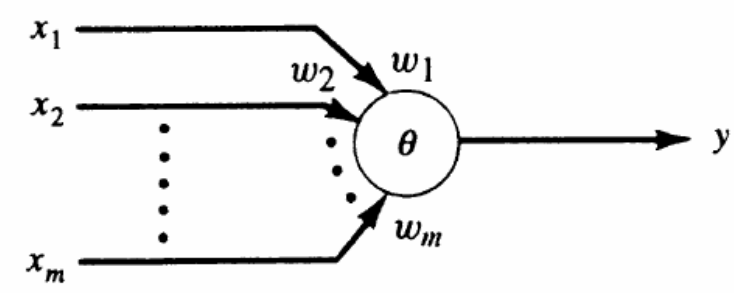



*FIGURE 2: McCullock and Pitts' artificial neuron [5].*

In the 1950's the mathematician John Von Neumann studied the generation of life-like behaviours in machines. One of his brilliant works showed how to build and automaton capable of self-reproduction. He used Cellular Automata with 29 states per cell as a representation of a life-like computation paradigm because of the bottom-up, parallel and locally determined behaviour, giving the proof that self-reproduction was achievable by machines. The information contained in the description of the machine was basically used in two very different ways: it was interpreted as instruction to build the new machine and it was uninterpreted as data to be duplicated and to be given to the offspring (it turned out that it is the same for DNA). Later, Codd in 1968 developed a new version of the self-replicating automata using only 8 states per each cell.

Many other studies in this field have been done until now (Conway's game of life, Ray's Tierra and so on), all trying to map natural phenomena into machines, generating complex dynamics. In all those works, the idea was to borrow principles from nature and to use it to build life-like behaviours. The most promising models were conceived without a central controller but with a distributed mechanism for the control of the behaviour.

In the next sections a more detailed description of the analogy between nature and machines and the current state of the art are given.

# Development and Evolution

## *Genotype and Phenotype*

In the previous chapter we have distinguished between the specification of the machine and the resulting machine's behaviour. In living systems there is the same distinction, where the genotype (the specification of the machinery) represents the genetic instructions and the phenotype (the behaviour of the machinery) is the organism that emerges from a process called morphogenesis, which consists of the interpretation of the genotype in a specific environment. The genotype is usually a very large set of instructions and each one is triggered and executed under specific conditions. In other words, the phenotype is the result of a **development process** which is a nonlinear function of the genotype. For this reason, it is very difficult to predict the specific phenotype that will emerge from a given genotype. It is also very unlikely to understand which modification should be done on the genotype in order to change a characteristic in the phenotype. The only method available is to try and check the results. This is also called **evolution by natural selection**, where the fittest element will survive and the weakest will die.

## *Example of Development: L-Systems*

Different approaches have been exploited to study developmental models. With cellular automata the cells can develop in parallel but the space is limited and the process can continue only until the edges of the cellular array are reached. With Chomsky grammars it is possible to generate longer expressions but not in parallel; in fact the rewriting rules are executed sequentially. With L-Systems both the requirements are satisfied. The structure can expand everywhere and also in parallel.

Formally an L-System is a triplet $G = < V, w, P >$ where V is an alphabet ($V^*$ represents the set of all words over V and $V+$ the set of all non empty words over V), $w \in V+$ is a non-empty word called axiom and $P \in V \times V^*$ is a finite set of productions. A production is a pair (a, X) where a is a letter and X is a word, we write $a \rightarrow X$. Starting from the axiom it is possible to generate a sequence of words applying the production rules. We say that a word w is derived from the axiom if exists a developmental sequence of words, generated using the production rules, that goes from the axiom to the word w. In a developmental sequence, each word has a predecessor and a successor.

There are two main types of productions that can be applied:

- Context-free productions, that can be applied regardless of the context in which the predecessor appears;
- Context-sensitive productions, where the context has to be taken into account. With context-sensitive L-Systems it is possible to model information exchange between cells. The context can be applied on both sides (2L-Systems) or only in one side (1L-Systems).

***Example of 1L-System:***

*w: baaaaaa*
*p: b < a → b*

*Generated words:*
*baaaaaa*
*bbaaaaa*
*bbbaaaa*
*.....*

L-Systems are particularly suitable for plant modelling. For this purpose, it is necessary to explicit the following definitions:

- Axial tree is a rooted tree where for each node there is only one straight segment and all the other edges are lateral;
- Axis is a sequence of segments where each of the segments is straight and the last one is not followed by any straight segment;
- Branch: an axis with all its descendants.

If an axis starts from the root, it has order zero. An axis originated from an n-order axis has order n+1. With this enumeration, all the axes and branches are ordered. Fig. 3 shows and example of tree.

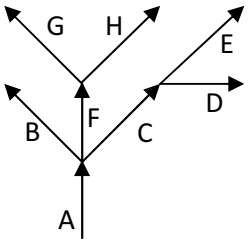



*FIGURE 3: Example, graphical representation of a tree [2].*

There are many representations of L-Systems, every one using specific symbols or techniques in order to model specific mechanisms, e.g., the growth of flowers on plants. In particular, square brackets are introduced to represent different branches and delays are implemented with special productions, e.g., some rules can be non deterministic and after the production of a certain number of leaves a flower will grow. In addition, stochastic mechanisms can be introduced, with production rules having different probabilities. Environmental change can be modelled with different sets of production rules and table changes according to some environmental factors. Another way of triggering some events such as for flowers production is signal propagation. Moreover, it is possible to extend the notion of L-System to a 2D or 3D structure in order to represent complex configurations.

L-Systems can be divided in several classes, depending on the type of language they can generate:

- OL-Systems: language class generated by a context-free L-System;
- IL-Systems: language class generated by a context-sensitive L-System;
- DOL-Systems: language class generated by a deterministic and context-free L-System;
- TOL-Systems: language class generated by a context-free L-System, but different rewriting steps may use different sets of production rules (tables).

All these classes of L-Systems have different levels of complexity and they can develop structures with specific mathematical properties. In order to prove that a certain class of L-Systems is not able to generate some sequence of structures, it is necessary to use a tool called "Pumping Lemma", where a language is a set of strings that may or may not be produced by a class of L-Systems.

The Pumping Lemma says that for a particular language to be a member of a certain class, any sufficient long string in the language contains a section, or sections, that can be removed or repeated any number of times, with the resulting string remaining in the language. The Pumping Lemma is a necessary but not sufficient condition for class membership.

In conclusion, with L-Systems it is possible to produce complex and dynamic development processes using few simple production rules, generating a graphical representation of the developing organism.

## *Example of Evolution: Genetic Algorithms*

Artificial Evolution is a process that reproduces the natural selection over a population of machines. In this context, some key principles have to be fulfilled in order to have a true artificial evolution process:

- Heredity: parents and offspring are similar, the copy of the genome to the next generation is accurate;
- Variability: even if the copy process has to be accurate it is not perfect so the offspring is not an exact copy of the parents;
- Fecundity: variation influences the behaviour of the new generation and the change of behaviour affects on the success of reproduction.

A Genetic Algorithm (GA) is an idealized computational version of Darwinian evolution. In Darwinian evolution, organisms reproduce at differential rates, with fitter organisms producing more offspring than less fit ones. Offspring inherit traits from their parents; those traits are inherited with variation via random mutation, sexual recombination, and other sources of variation. Thus traits that lead to higher reproductive rates get preferentially spread in the population, and new traits can arise via variation.

In GAs, computer "organisms" (usually encoded as bit strings) reproduce in proportion to their fitness in the environment. Offspring inherit traits from their parents with variation coming from random

mutation, in which parts of an organism are changed at random, and sexual reproduction, meaning that an organism is made up of recombined parts coming from the chromosomes of its parents.

An Evolutionary Algorithm (EA) is often composed by the following steps:

- Generation of a random initial population of individuals.
- **Evaluation** of the "quality" of the phenotypes in the population through a Fitness Function. The level of fitness indicates the probability of success of the phenotype in the environment;
- **Selection** of high quality individuals in the population (high fitness means high probability of being chosen for the reproduction process);
- **Crossover**: recombination of the genotypes from the chosen phenotypes to produce offspring. This step can be done with different methods, an example is shown in Fig. 4;
- **Mutation**: random modification of small parts of the new genotype with low probability (mutation rate).

This process is iterated for many generations, at which point hopefully one or more high-fitness individuals have been created.

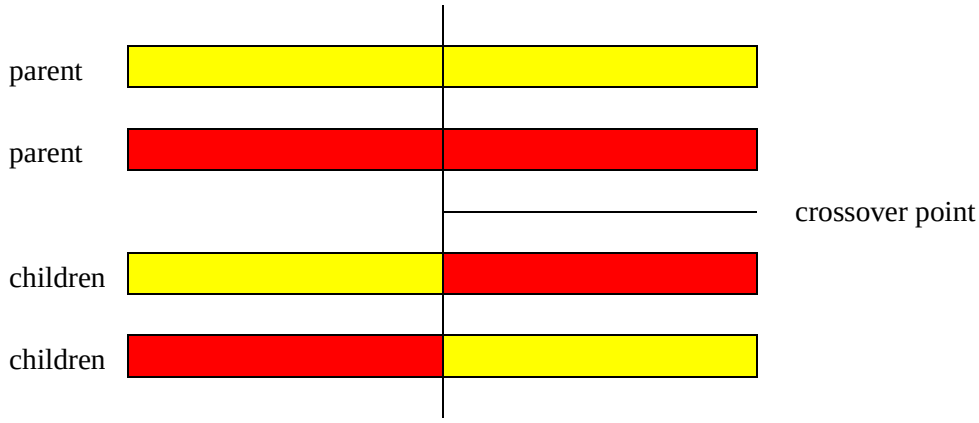



*FIGURE 4: One-point uniform crossover: a single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children.*

Summarizing, what a GA does is to explore a very large phenotype solution space in a clever manner. In other words, it is an attempt to develop a new computational paradigm inspired by a nature and biology, a life-like non-linear process. Non-linear in the sense that the whole behaviour of the system is not just the sum of the behaviours of its parts but is the behaviour that emerges from the local interaction of its parts. If those small parts of the system are studied independently, those emergent properties disappear.

*"Neither nucleotides nor amino acids nor any other carbon-chain molecule is alive – yet put them together in the right way, and the dynamic behaviour that emerges out of their interactions is what we call life. [...] life is a kind of behaviour, not a kind of stuff."[1].*

# Cellular Automata

Cellular automata, originally studied by Ulam [8] and von Neumann [9] in the 1940s, are idealized versions of parallel and decentralized computing systems, based on a myriad of small and unreliable components called cells. Even if a single cell itself can do very little, the emergent behavior of the whole system is capable to obtain complex dynamics. In cellular computing each cell can only communicate with a few other cells, most or all of which are physically close by (neighbors). One implication of this principle is that no one cell has a global view of the entire system, there is no central controller. The metaphor with biology can be exploited on cellular systems because the physical structure is similar to the biological multi-cellular organisms.

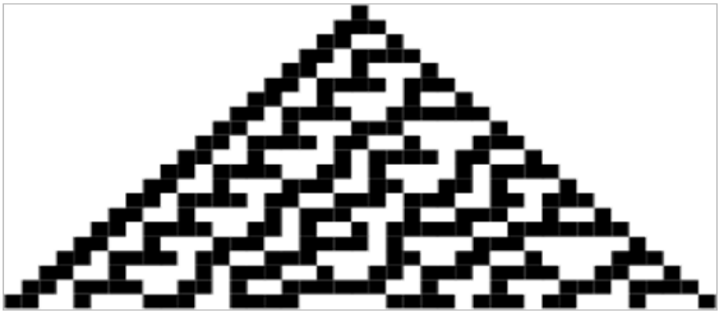
Formally, a cellular automaton consists of a countable array of discrete sites or cells $i$ and a discrete-time update rule $\Phi$ operating in parallel on local neighborhoods of a given radius $r$. At each time the cells take on values in a finite alphabet $A$ of primitive symbols: $\sigma^i_t \in \{0, 1,..., k-1\} \equiv A$. The local site-update function is written:

$$\sigma^i_{t+1} = \Phi(\sigma^{i-r}_t, \ldots., \sigma^{i+r}_t)$$

The state $s_t$ of the CA at time $t$ is the configuration of the finite or infinite spatial array: $s_t \in A^N$, where $A^N$ is the set of all possible cell value configurations on a lattice of $N$ cells. The "extended state space", denoted $A^*$, is the union of all states of any $N$:

$$A^* = \mathrm{U}_{N \geq 0} A^N \qquad \text{with} \qquad A^0 = \emptyset.$$

The CA global update rule $\Phi$: $A^N \rightarrow A^N$ applies $\Phi$ in parallel to all sites in the lattice: $s_t = \Phi\, s_{t-1}$. For finite $N$ it is also necessary to specify a boundary condition. The boundary cells are dealt with by having the whole lattice wrap around into a torus, thus boundary cells are connected to “adjacent" cells on the opposite boundary. In Fig. 5 an example of 1D CA development is shown.



| Neighboorhood | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| Next cell value | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

$\sigma^{j-1}_t\, \sigma^j_t\, \sigma^{j+1}_t$

$\sigma^j_{t+1}$

*FIGURE 5: 1D CA developed with rule 30 [13], represented in binary by 00011110*

## *Example of CA properties: Evolving a Self-Repairing, Self-Regulating French Flag Organism*

Cellular automata can be used to abstract and simulate a biological development of an organism. Miller and Banzhaf tried to exploit self-repair and self-regulation properties on CAs in order to create a "2D French Flag Organism" able to grow until a certain size and remain stable afterwards. To do so, they built a cellular world where each cell was carrying the same genotype and the possible actions of each cell were growing, dying, changing colour or releasing chemicals. It is important to say that there is no relationship between the genotype size and the size of the phenotype.

In their tests, each cell was able to read its own state and the state of the other 8 neighbours and to see the amount of chemicals that are present on each of those cells. Chemicals are responsible for the type of action that each cell will perform at every time-step. With the chemical information, the cell program will decide if the cell has to duplicate, die, change or produce a certain amount of chemicals to release in the environment. Moreover, the cell genotypes are evolved and the phenotypes are confronted with the desired result. A genetic algorithm then evolves the genotypes to produce the new population of cellular programs and eventually, after a certain number of iterations, the target organism will emerge. In Figure 6 the growth process is shown.

Another test investigated the behaviour of the growing organism, removing parts of it (a diagonal section or a rectangular section). The results showed that the embryonic arrays were able to repair themselves after damage. It turned out that this process of self-repair and self-regulation was much faster with chemicals. Miller stated that *"having chemicals makes it easier to find fitter organisms, and the result indicate that having either none or one chemical make it unlikely, if not impossible, to achieve solutions that grow and then stop growing that meet the target objective" [10].*

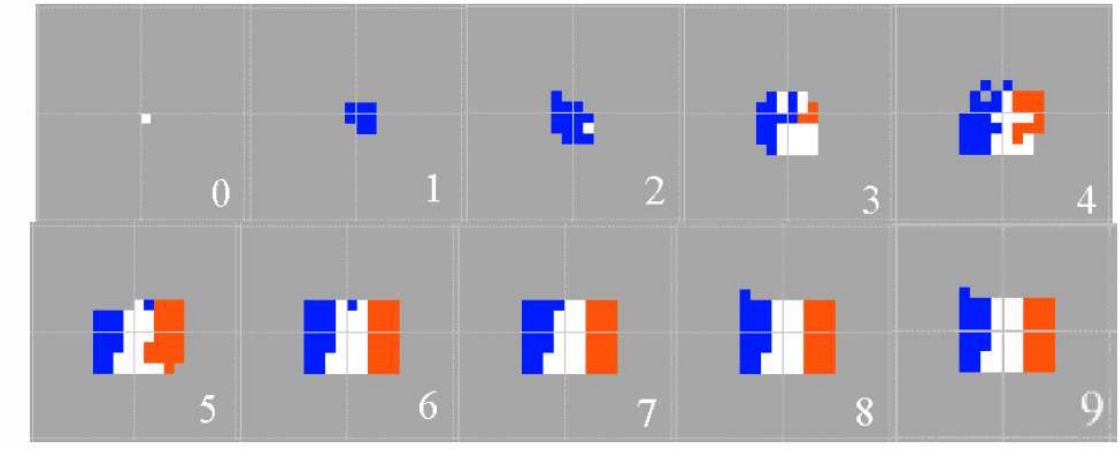



*FIGURE 6: Example of a French Flag Organism growing from a zygote (step 0, white cell) [4]*

## *CA Classes*

Cellular automata can be used to model different types of complex systems. Each of the rules can produce very simple or complex patterns. According to Stephen Wolfram [12], all the possible CA behaviours can be grouped in 4 classes:

- Class 1 CA: after a few steps an homogeneous state of equilibrium (a point attractor) is reached, independently of the initial conditions;
- Class 2 CA: after a few time steps a constant periodic pattern is reached. Few specific positions of the pattern may be dependent on some initial conditions but not the general behaviour;
- Class 3 CA: the produced patterns seem to be random and irregular;
- Class 4 CA: locally complex structures may appear.

Whilst class 1 and class 2 automata do not preserve the initial information, the dynamics of classes 3 and 4 are highly correlated to the initial conditions. Those behaviours can resemble, to a certain degree, the actions that happen to molecules in materials like liquids, solids, gases or the behaviour of living organisms. For those reasons, specific types of cellular automata can be used to model some laws of physics or thermodynamics. From ordered and simple initial conditions, according to the second law of thermodynamics, the entropy of a system (disorder and randomness) increases and irreversibility is quite probable (but not impossible, as stated by Poincaré's theorem of reversibility [11]). One example of such reversible automaton is shown in Figure 7.

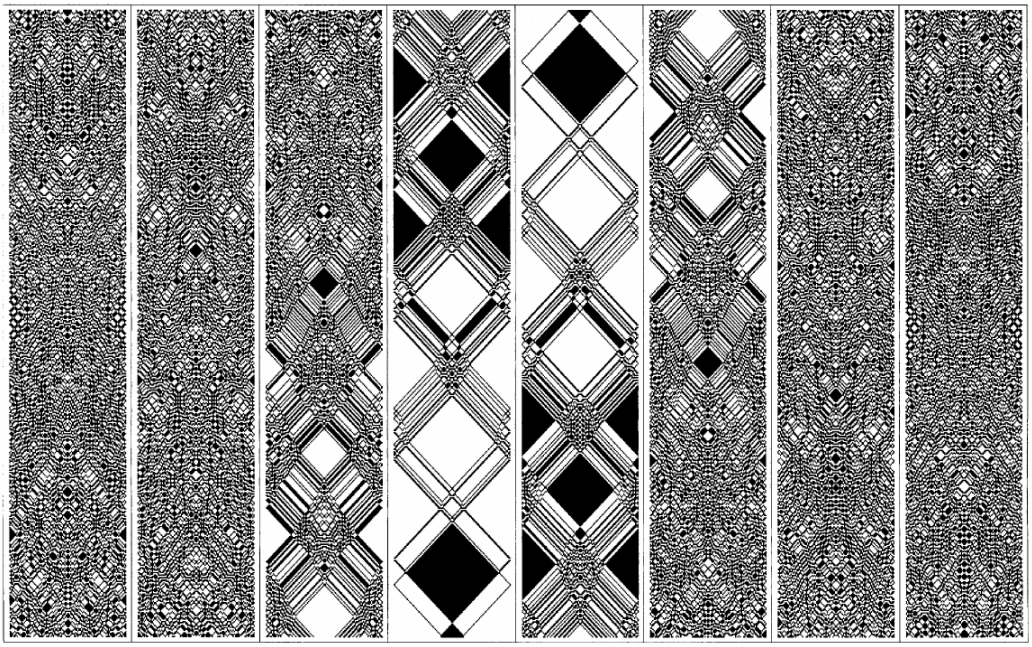


*FIGURE 7: Reversible cellular automaton with random pattern formation illustrating the 2nd law of thermodynamics [5]*

# Summary

In this paper a notion of Artificial Life is presented. One of the key points is to reproduce the behaviour of living systems using computers. This is often done by building an artificial population of elements and let them interact on local basis. The whole global behaviour, than, has to be analyzed as an emergent property that seem intelligent and alive.

What we call nowadays Artificial Life originated in the ancient eras with Egyptians but only recently, in 1735, the first complex mechanical mechanism was developed by Jacques de Vaucanson. In 1943 McCullock and Pitts modelled an abstraction of an artificial neuron but the most revolutionary studies in the field have been conducted by John Von Neumann in the '50s. Ever since, the field of Artificial Life has grown rapidly with many contributions from all over the world.

The distinction from genotype and phenotype, that is present in biology, has been exploited in machines using artificial development and artificial evolution. Some examples are presented: L-Systems as an artificial development paradigm and Genetic Algorithms as model of artificial evolution. Cellular Automata, idealized versions of parallel and decentralized computing systems, have been presented together with an example of emergent properties such as self-repair and self-regulation. The behaviour of all CA can be grouped in four sets, according to Wolfram' studies on the produced patterns.

Richard Dawkins in 1986 wrote in his book The Blind Whatchmaker that "you cannot get out of computers any more than you put in, that computers only do exactly what you tell them to, and that therefore computers are never creative. The cliché is true only in the crashingly trivial sense, the same sense in which Shakespeare never wrote anything except what his first schoolteacher taught him to write--words".

Man is becoming closer to machines and machines are becoming closer to man. Eventually, these technologies will become so powerful that we will use them to create life artificially. This can have hazardous consequences. And yet, I believe that this voyage of discovery will lead us to a better life, and that's why I want to offer my small scientific contribution.

# Bibliography

*[1] – Langton, C. - Artificial Life, pag. 1-48 - Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87), Los Alamos, NM, USA, September 1987. Santa Fe Institute Studies in the Sciences of Complexity 6 Addison-Wesley 1989, ISBN 0-201-09346-4*

*[2] – Lindenmayer, A. - Developmental Models of Multicellular Organisms: A Computer Graphics Perspective, pag. 221-250 - Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87), Los Alamos, NM, USA, September 1987. Santa Fe Institute Studies in the Sciences of Complexity 6 Addison-Wesley 1989, ISBN 0-201-09346-4*

*[3] – Langton, C. - Computation at the Edge of Chaos: phase transition and emergent computation, pag. 12-37 - Physica D 42 1990 — Elsevier Science Publisher B.V. (North-Holland)*

*[4] – Kumar, S. and Bentley, P. - On Growth, Form and Computers - Elsevier Academic Press, Year 2003 – 444 pages – ISBN 0-12-428765-4*

*[5] – Mainzer, K. - Thinking in Complexity – Springer, Year 1994 – 482 pages – ISBN 978-3-540-72227-4*

*[6] – Turing, A. - On computable numbers, with an application to the Entscheidungsproblem, - Proceedings of the London Mathematical Society, Series 2, 42 pp 230–265, 1936*

*[7] – McCulloch, W. S. and Pitts, W. H. - A logical calculus of the ideas immanent in nervous activity. - Bulletin of Mathematical Biophysics, 5:115-133, 1943*

*[8] – Ulam, S. - Los Alamos: Los Alamos Science, 1987. Vol. 15, pp. 1-318, special issue. -Los Alamos National Laboratory, 1909-1984*

*[9] – Von Neumann, J. - Theory and Organization of complicated automata. A. W. Burks, 1949, pp. 29-87 [2, part one]. Based on transcript of lectures delivered at the University of Illinois in December 1949*

*[10] – Miller, J. - Evolving a Self-Repairing, Self-Regulating, French Flag Organism - Gecco 2004. Springer-Verlag Lecture Notes in Computer Science 3102, pp. 129-139*

*[11] – Poincaré, H. - Sur le probleme des trois corps et les equations de la dynamique - Acta Math. 13, pp. 1-270, 1890*

*[12] – Wolfram, S. - Universality and Complexity in Cellular Automata. Physica D Volume 10 Issue 1-2, pp. 1-35, 1984*

*[13] – Wolfram, S. - A New Kind of Science. Wolfram Media Inc, Year 2002 – 1197pages – ISBN 1-57955-008-8*