Fifty Years of Research on Self-Replication: An Overview

Moshe Sipper
Logic Systems Laboratory
Swiss Federal Institute of Technology
IN-Ecublens
CH-1015 Lausanne
Switzerland
Moshe.Sipper@di.epfl.ch
http://lslwww.epfl.ch/~moshes

Abstract The study of artificial self-replicating structures or machines has been taking place now for almost half a century. My goal in this article is to present an overview of research carried out in the domain of self-replication over the past 50 years, starting from von Neumann's work in the late 1940s and continuing to the most recent research efforts. I shall concentrate on computational models, that is, ones that have been studied from a computer science point of view, be it theoretical or experimental. The systems are divided into four major classes, according to the model on which they are based: cellular automata, computer programs, strings (or strands), or an altogether different approach. With the advent of new materials, such as synthetic molecules and nanomachines, it is quite possible that we shall see this somewhat theoretical domain of study producing practical, real-world applications.

Keywords
self-replication, cellular automata, self-replicating programs, self-replicating strings, self-replicating machines

1 Introduction

In the late 1940s eminent mathematician and physicist John von Neumann had become interested in the question of whether a machine can self-replicate, that is, produce copies of itself. Von Neumann wished to investigate the *logic* necessary for replication—he was not interested in, nor did he have the tools for, building a working machine at the biochemical or genetic level. Remember that at the time DNA had not yet been discovered as the genetic material in nature.

The study of artificial self-replicating structures or machines has been taking place now for almost half a century. Much of this work is motivated by the desire to understand the fundamental information-processing principles and algorithms involved in self-replication, independent of their physical realization. An understanding of these principles could prove useful in a number of ways. It may advance our knowledge of biological mechanisms of replication by clarifying the conditions that any self-replicating system must satisfy and by providing alternative explanations for empirically observed phenomena. The fabrication of artificial self-replicating machines can also have diverse applications, ranging from nanotechnology [22, 51] to space exploration [27].

My goal in this article is to present an overview of research carried out in the domain of self-replication over the past 50 years, starting from von Neumann's work. I shall concentrate on computational models, that is, ones that have been studied from a computer science point of view, be it theoretical or experimental. I shall only briefly touch upon other types of works (e.g., involving self-replicating molecules).

All the works described herein are represented in Figure 1, delineating both the chronological development and the conceptual lineage. Furthermore, the figure classifies them into four classes, according to the model on which they are based: cellular au-

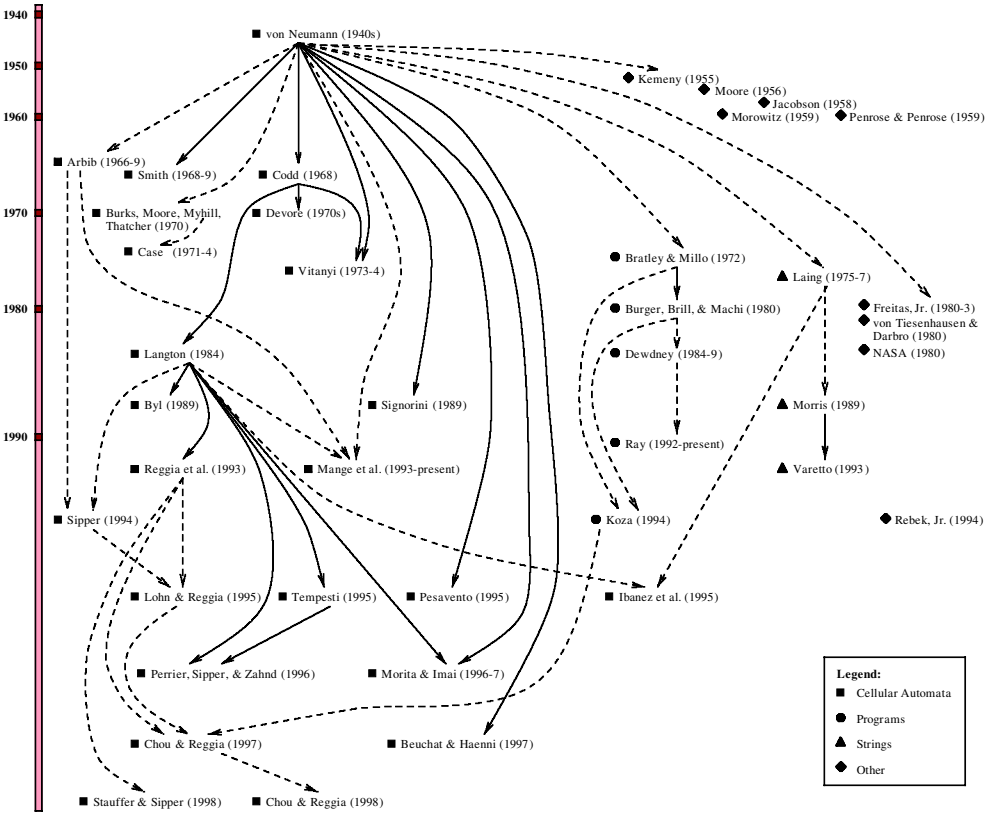


Figure 1. Lineage of works in the area of self-replication. A solid line represents a close relationship between two works, e.g., the later work may be an extension or implementation of the earlier one. A dashed line represents a conceptual relationship between two works, e.g., the later work is based on some of the ideas laid down in the earlier one.

tomata, computer programs, strings (or strands), or an altogether different approach. As can be seen, a majority of the works are based on the cellular-automaton model, the one originally used by von Neumann. The model, as well as the works that have employed it, is described in the next section, followed by program-based approaches (Section 3), string-based works (Section 4), and ending with a brief description of a few other systems (Section 5). Finally, I conclude in Section 6. The interested reader might also wish to consult the online self-replication page at http://lslwww.epfl.ch/~moshes/selfrep/, which includes—in addition to information on self-replicating systems—links to online sites and demos.

Before proceeding I wish to note the following point concerning terminology: In a recent paper, Sipper et al. [75] made a distinction between two terms, *replication* and *reproduction*, which are often considered synonymous. Replication is an ontogenetic, that is, developmental process, involving no genetic operators, resulting in an exact duplicate of the parent organism. Reproduction, on the other hand, is a phylogenetic, that is, evolutionary process, involving genetic operators such as crossover and mutation, thereby giving rise to variety and ultimately to evolution. In most works described herein these two terms are considered synonymous and are used interchangeably (indeed, most researchers seem to have opted for the somewhat less correct term of reproduction).

2 Cellular Automata-Based Works

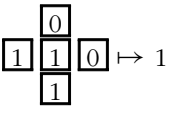
The following section describes research into self-replication that is based on the cellular automaton (CA) model. I first present the model itself (Section 2.1), followed by a description of the works that employ it, divided into four categories: research carried out within the framework of von Neumann's CA model (Section 2.2), simple self-replicating structures (Section 2.3), self-replicating structures with added computational capabilities (Section 2.4), and finally, works in which the underlying CA model has been altogether modified (Section 2.5).

2.1 Cellular Automata

One of the central models used to study self-replication is that of cellular automata. Indeed, they were originally conceived by Ulam and von Neumann in the 1940s with this objective in mind—to provide a formal framework for investigating the self-replication issue. CAs are dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells.

The cellular array (grid) is n-dimensional, where $n = 1, 2, 3$ is used in practice; in this article I shall concentrate on $n = 2$, that is, two-dimensional grids. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table (also known as the transition function), with an entry for every possible neighborhood configuration of states. The cellular neighborhood of a cell consists of the surrounding (adjacent) cells. For one-dimensional CAs, a cell is connected to r local neighbors (cells) on either side, as well as to itself, where r is a parameter referred to as the radius (thus, each cell has $2r + 1$ neighbors). For two-dimensional CAs, two types of cellular neighborhoods are usually considered: five cells, consisting of the cell along with its four immediate nondiagonal neighbors (also known as the von Neumann neighborhood), and nine cells, consisting of the cell along with its eight surrounding neighbors (also known as the Moore neighborhood). When considering a finite-sized grid, spatially periodic boundary conditions are frequently applied, resulting in a circular grid for the one-dimensional case, and a toroidal one for the two-dimensional case. For a formal definition of CAs, as well as additional material, the reader is referred to [31, 50, 73, 86].

As an example, let us consider the parity rule (also known as the XOR rule) for a two-state, five-neighbor, two-dimensional CA [73]. Each cell is assigned a state of 1 at the next time step if the parity of its current state and the states of its four neighbors is odd, and is assigned a state of 0 if the parity is even (alternatively, this may be considered a modulo-2 addition). The rule table consists of entries of the form

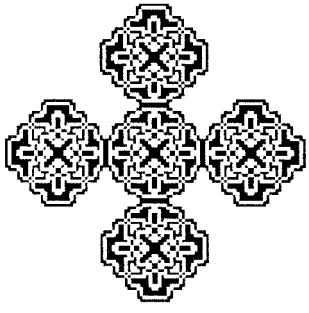


This means that if the current state of the cell is 1 and the states of the north, east, south, and west cells are 0,0,1,1, respectively, then the state of the cell at the next time step will be 1 (odd parity). The rule is completely specified by the rule table given in Figure 2a. Figure 2b shows a pattern produced by iterating this CA rule.

One of the reasons for the ubiquitous use of CAs as a vehicle for studies in self-replication seems to be historical, due to von Neumann's choice of model. However,

CNESW	S_{next}	CNESW	S_{next}
00000	0	10000	1
00001	1	10001	0
00010	1	10010	0
00011	0	10011	1
00100	1	10100	0
00101	0	10101	1
00110	0	10110	1
00111	1	10111	0
01000	1	11000	0
01001	0	11001	1
01010	0	11010	1
01011	1	11011	0
01100	0	11100	1
01101	1	11101	0
01110	1	11110	0
01111	0	11111	1

(a)



(b)

Figure 2. Cellular automata: the parity rule. The cellular space is two-dimensional, two-state, with a five-cell neighborhood. (a) Parity rule table. CNESW denotes the current states of the center, north, east, south, and west cells, respectively. S_{next} is the cell's state at the next time step. (b) Starting from a 20 $\times$ 20 grid of cells in state 1, with all other cells being in state 0, the above pattern is produced after 90 time steps.

his decision was by no means arbitrary: CAs exhibit both simplicity and rigor—one can create an environment, or "universe," as it is sometimes referred to, using simple basic ingredients in a model that is mathematically rigorous.

2.2 Works Within the Framework of von Neumann's Model

As noted above, von Neumann had become interested in the issue of self-replication in the late 1940s, and following the suggestion of his colleague mathematician Stanislaw Ulam had adopted the CA model to study this problem [84]. Von Neumann used two-dimensional CAs with 29 states per cell and a five-cell neighborhood. He showed that a universal computer can be embedded in such cellular space, namely, a device whose computational power is equivalent to that of a universal Turing machine [33]. He also described how a universal constructor can be built, namely, a machine capable of constructing, through the use of a "constructing arm," any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, that is, of self-replicating (Figure 3). The terms "machine" and "tape" refer here to configurations, that is, patterns of states over the grid (indeed, the ability to describe such structures formally is a major advantage of CAs). Note that self-replication is obtained as a special case of universal construction, by having the constructor's artificial "genome" (i.e., input tape) contain a description of a universal constructor. This is quite a complex way of setting about if one wishes to attain but self-replication: While universal construction may represent a *sufficient* condition for attaining self-replication, it is by no means a *necessary* one, an observation that came to the fore with Langton's work, described in Section 2.3.

A noteworthy distinction apparent in von Neumann's model of self-replication is the double-faceted use of the information stored in the artificial genome: It first serves as instructions to be *interpreted* so as to construct a new universal constructor, after which this same genome is *copied* unmodified, to be attached to the new offspring

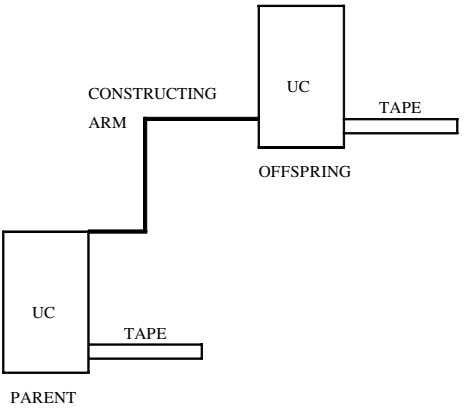
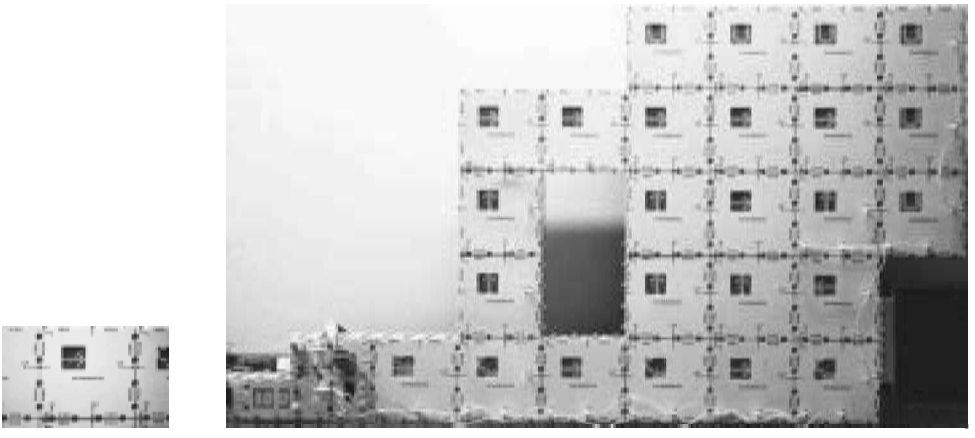


Figure 3. A schematic diagram of von Neumann's self-replicating cellular automaton. The system is a universal constructor (UC), namely, a machine (i.e., CA-embedded structure) capable of constructing, through the use of a "constructing arm," any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self-replicating. (The machine is not drawn to scale.)

constructor—so that it may replicate in its turn. This aspect is quite interesting in that it bears strong resemblance to the genetic mechanisms of transcription (copying) and translation (interpretation) employed by biological life—which were discovered during the decade following von Neumann's work.

Von Neumann's model employs a complex transition rule, with the total number of cells composing the universal constructor estimated at between 50,000 and 200,000 (the literature seems to disagree on the exact number). In the years that followed its introduction a number of researchers had worked toward simplifying this system. In the late 1960s Codd [16] reduced the number of states required for a self-replicating universal constructor-computer from 29 to 8. His self-replicating structure comprised about 100,000,000 cells [38]. A few years later Devore [17] simplified Codd's system, devising a self-replicating automaton comprising about 100,000 cells [38]. Concurrently, Smith [76] had opted for a different route: He noted that while von Neumann's demonstration of the possibility of machine self-replication involved a book-length *constructive* proof, a much shorter (two-page) *existence* proof could be had. Furthermore, whereas von Neumann required both computation universality and construction universality of his self-replicating machines, Smith [76] showed that computation universality alone suffices. He noted that "the proof here reduces the problem of self-construction to a computation problem, which means that no machinery beyond ordinary computation theory is required for self-reproduction" (p. 710).

Despite the complexity of von Neumann's self-replicating universal constructor, a number of researchers have considered its implementation (or simulation) over the years. Signorini [69, 70] concentrated on the 29-state transition rule, discussing its implementation on a SIMD (single-instruction multiple-data) computer. Von Neumann's constructor is divided into many functional blocks known as organs. In addition to implementing the transition rule, Signorini also presented the implementation of three such organs: a pulser, a decoder, and a periodic pulser. To date, Pesavento's more recent work comes closest to a full simulation of von Neumann's model [62]. A computer simulation of the universal constructor—running on a standard workstation—even this comes short of realizing the full model: Self-replication is not demonstrated because the tape required to describe the constructor (i.e., the genome) is too large to simulate.



(a) (b)

Figure 4. (a) Hardware implementation of the von Neumann cell using field-programmable gate arrays (FPGAs): top face of the von Neumann module, including connection points to other cells and an LED display showing the current state of the cell. (b) Hardware implementation of one of the organs of von Neumann's universal constructor, known as a pulser, using the module shown in (a). The above pulser P(11001) generates at the output cell (top right) the sequence of excitations (signals) 11001, a fixed number of time steps after receiving an excitation (i.e., a 1 signal) at the input cell (bottom left). Note that the 25 von Neumann modules are not arranged as a 5×5 square—in fact, the arrangement is that of a 7×7 square, where unused cells are simply not implemented. This allows for the construction of a larger organ for the price (literally) of a smaller one.

Interestingly, Pesavento used three more states per cell as compared with von Neumann (32 vs. 29), which resulted in a substantially smaller constructor.

A recent addition to these implementation efforts is the work of Beuchat and Haenni [6, 74], who constructed a hardware module that implements a single 29-state cell of von Neumann's model. Each module is embedded in a plastic box whose top face contains a number of connection points and an LED display showing the current state of the cell. Several such modules can be fitted together to produce a small cellular array. The sides of the modules contain electrical contacts, which allow adjacent cells to transmit information to each other without additional wiring. The cells were implemented using the recent technology of reconfigurable processors, specifically, field-programmable gate arrays (FPGAs) [50, 80]. To date, Beuchat and Haenni have used this module to implement a 25-cell organ, known as a pulser. For example, a pulser P(11001) generates at a designated output cell the sequence of excitations (signals) 11001, a fixed number of time steps after receiving an excitation (i.e., a 1 signal) at a designated input cell. The machine is shown in Figure 4.

Going back to 1970, Burks [9] had edited a compendium of works on cellular automata, in particular drawing inspiration from von Neumann's work (Burks was a close friend and colleague of von Neumann, and in fact had completed, posthumously, the latter's work on self-replication [84]). Of special interest are the first chapter: "Von Neumann's self-reproducing automata," written by Burks himself; Chapter 4: "Self-describing Turing machines and self-reproducing cellular automata," by Thatcher; Chapter 6: "Machine models of self-reproduction," by Moore; and Chapter 8: "The abstract theory of self-reproduction," by Myhill.

In the early 1970s, partly inspired by Myhill's chapter in Burks's book, Case considered machines that construct distortions of themselves [11–13]. This includes the cases of machines that eventually have a sterile descendant, those that after a delay of m generations repeat every n generations, and those that are aperiodic over generations.

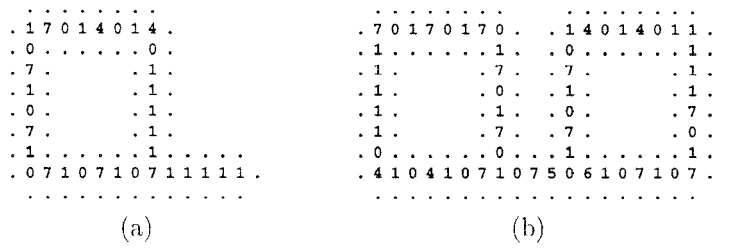


Figure 5. Langton's self-replicating loop. (a) Time step 0. (b) Time step 126. Sheath cells are denoted by dots. (See Figure 6 for demonstration of a full replication cycle of a similar loop.)

He also discussed the biological meaning of periodicity over generations, where the period n is greater than one. It was shown that there are no algorithms for deciding of a progenitor machine many properties of its descendants. For example, there is no algorithm for deciding of a progenitor that does not have sterile descendants whether its descendants are periodic or aperiodic in subsequent generations.

Finally, also in the 1970s, Vitányi [81–83] modeled sexual reproduction within von Neumann's formal framework. He argued that the transition from asexual to sexual reproduction necessitates a change in the number and structure of the genetic tapes involved (the artificial genomes). To an asexually reproducing automaton only one genetic tape is attached, that is, the description that enables the automaton to construct cell for cell a replica of itself. The sexually reproducing automaton, however, must possess two, nearly identical, genetic tapes of a deviating structure, that is, programs partitioned into sections embodying the various construction and behavioral algorithms to be executed. Vitányi showed that the recombination of the parents' characteristics in the offspring closely conforms to recombination in nature. He also discussed similarities and differences with biological systems.

2.3 Simple Self-Replicating Structures

In 1984, Langton observed that although the capacity for universal construction, as discussed in the previous subsection, is a *sufficient* condition for self-replication, it is not a *necessary* one. Furthermore, he remarked that natural systems are probably not capable of universal construction. Langton therefore set out to discover what kind of logical organization is necessary for an automaton to be able to replicate itself; that is, he sought a small structure that could accomplish but one task—self-replication [42, 43].

Langton's self-replicating structure is a loop constructed in a two-dimensional, eight-state, five-neighbor cellular space and is based on one of Codd's elements, known as a periodic emitter [16] (Figure 5). The 86-cell loop is basically a closed data path, consisting of a string of core cells in state 1, surrounded by sheath cells in state 2 (this latter state is represented by dots in Figure 5). Data paths are capable of transmitting data in the form of signals, which are packets of two cotraveling states: the signal state itself (state 4, 5, 6, or 7) followed by the state 0. The signals contained within the loop cycle through it, composing the instructions for replication, that is, the artificial genome. As each such signal encounters the arm junction it is duplicated, with one copy propagating back around the loop again and the other copy propagating down the arm, where it is translated as an instruction when it reaches the end of the arm. In executing the instructions the arm extends itself and folds, ultimately resulting in a daughter loop, also containing the genome needed to replicate.

As did von Neumann, Langton emphasized the two different modes in which in-

formation is used, interpreted (translation) and uninterpreted (transcription). In his loop, translation is accomplished when the instruction signals are executed as they reach the end of the construction arm, and upon collision of signals with other signals. Transcription is accomplished by the duplication of signals at the arm junctions.

Langton's loop, being stripped of any construction capabilities, is strikingly simple, can be easily simulated, and accomplishes its intended task: self-replication. Following in his footsteps, Byl [10] devised a smaller self-replicating loop, comprising 12 cells and embedded in a six-state cellular space. Reggia et al. [67] constructed yet smaller loops, sheathed as well as unsheathed, the smallest comprising just five cells, embedded in a six-state cellular space. They also studied cellular spaces exhibiting both weak and strong rotational symmetry (briefly, weak rotational symmetry means that some cell states are directionally oriented while with strong rotational symmetry all cell states are viewed as being unoriented [67]).

Ibánez et al. [34] devised a number of loops in a 16-state cellular space. Replication in their system is based not on having a fixed artificial genome, undergoing both translation and transcription, but rather on *self-inspection*, where the description of the object to be replicated (the genome) is dynamically constructed concomitantly with its interpretation. I will return to this point in Section 4. Another interesting property of their approach concerns the fact that the loops are not necessarily square ones as with Langton-like loops.

Morita and Imai [55–57] designed simple self-replicating loops in reversible cellular automata. A reversible cellular automaton is a special type of CA in which every grid configuration of states has at most one predecessor. Roughly speaking, it is a "backward-deterministic" CA. A prime motivation for studying such systems stems from the observation that computers based on reversible logic can be more efficient (e.g., faster, smaller, larger memories, lower energy consumption) [4, 5].

Stauffer and Sipper [77] observed that though CAs have been ubiquitously used over the years to study the issue of self-replication, the L-systems model [44, 63] is naturally suited for modeling growth processes, of which replication is a special case. They showed that L-systems can be used to specify self-replicating structures and explored the relationship between L-systems and CAs. The loop they devised—a small one, similar to those of Reggia et al. [67]—is depicted in Figure 6. Stauffer and Sipper concluded that the bridge between CAs and L-systems seems to offer a promising approach in the study of self-replication, and, more generally, of growth processes in CAs.

The self-replicating structures described up to this point were all designed by hand, a difficult and time-consuming process. Lohn and Reggia [46, 47] and Lohn [45] used genetic algorithms [52, 53] to discover automatically automata rules that govern emergent self-replicating processes. Given dynamically evolving automata, one of the most difficult tasks is that of identifying effective fitness functions for self-replicating structures. Lohn and Reggia were able to solve the fitness problem, thereby demonstrating that self-replicating structures can be evolved. Chou and Reggia [14] asked whether self-replication can come about spontaneously. They demonstrated the possibility of creating a CA universe—a "primordial soup"—in which self-replicating structures are not inoculated *ab initio* but rather emerge in a spontaneous manner (cf. Koza's work—Section 3). Their CA model incorporates a number of interesting features, including (a) the division of the cellular state into substates (bit fields), which facilitates the emergence of self-replication (Morita and Imai [55–57] presented a similar idea—the partitioned CA), (b) support of extended replication, in which the offspring structure may differ in size from its parent, and (c) the ability to handle collisions between structures, a situation that in previous models usually led to utter chaos. These

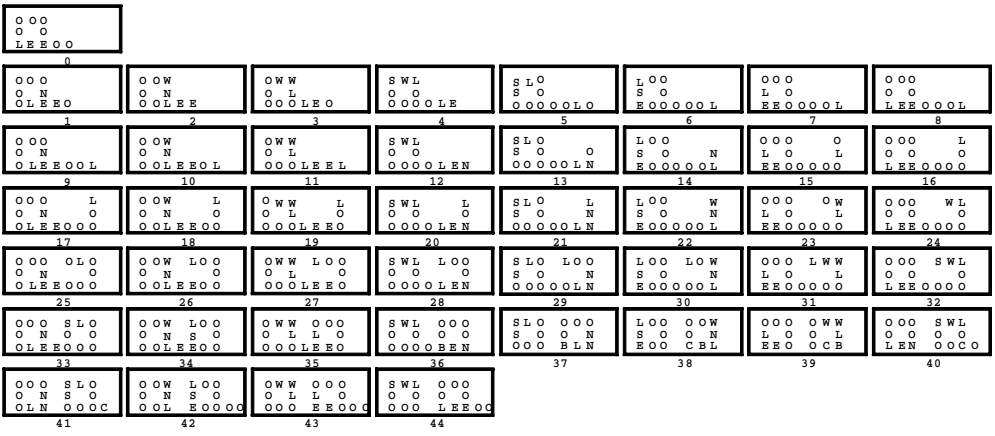


Figure 6. A self-replicating loop devised by Stauffer and Sipper [77]. The cellular space is two-dimensional, five-neighbor, with nine states per cell. Numbers at bottom of images denote time steps. The initial loop at time step 0 is replicated after 44 time steps.

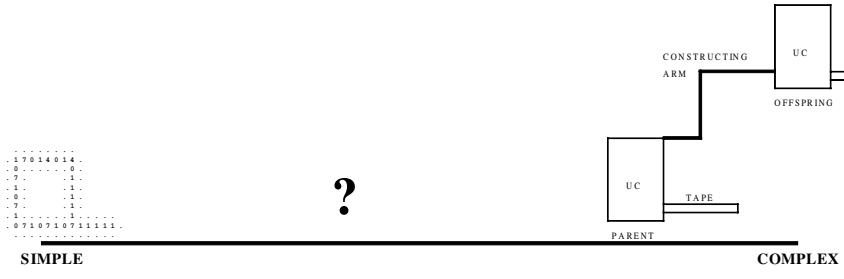


Figure 7. The complexity scale of self-replicating structures in cellular automata. On one end there are the highly complex universal constructor-computers, and on the opposite end one finds the simple structures that can do nothing but self-replicate. What kinds of structures can be devised in between the two?

qualities render their CA somewhat more robust than previous models and give rise to interesting dynamics.

2.4 Adding Computational Capabilities to Self-Replicating Structures

The previous two subsections presented works that can be considered to occupy two extremes of the replication complexity scale: on one end there are the highly complex universal constructor-computers, and on the opposite end one finds the simple structures that can do nothing but self-replicate (Figure 7).

Tempesti [78] asked whether one can start at the low end of the complexity spectrum, namely, with simple self-replicating structures, and *add* functionalities to these entities, ultimately attaining complex machines that are nonetheless completely realizable. He took a first step in this direction, devising a self-replicating system resembling that of Langton's—with the added capability of attaching to the automaton an executable program that is duplicated and executed in each of its copies. The program is stored within the loop, interlaced with the replication code (Figure 8).

Perrier, Sipper, and Zahnd [61] went beyond Tempesti's demonstration of finite computation, constructing a self-replicating loop that is capable of implementing any program, written in a simple yet universal programming language. The system consists of three parts—loop, program, and data—all of which are replicated, followed by the

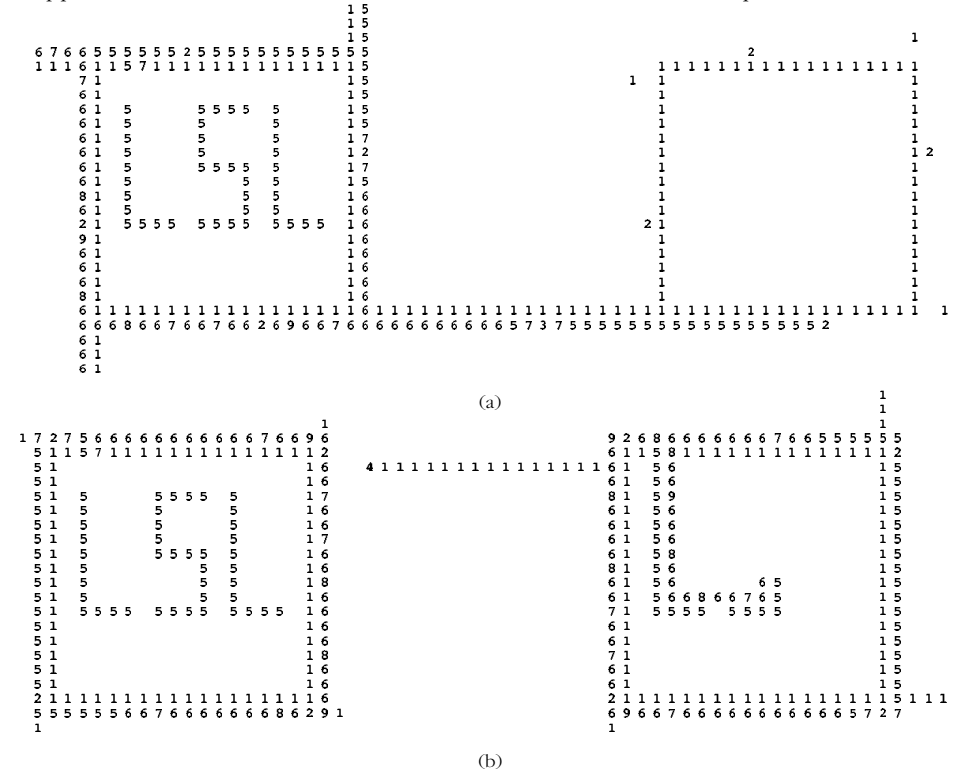


Figure 8. Tempesti's loop is a self-replicating automaton, with the added capability of attaching an executable program that is duplicated and executed in each of its copies. This is demonstrated above for a simple program that writes out (after the loop's replication) LSL, acronym of the Logic Systems Laboratory. (a) Time step 240: the program is being copied into the daughter loop. (b) Time step 341: the program is being executed in the daughter loop. The cellular space is two-dimensional, nine-neighbor, with 10 states per cell.

program's execution on the given data. Their loop was simulated in its entirety, thus demonstrating a viable, self-replicating machine with programmable capabilities (Figure 9).

Chou and Reggia [15] have recently shown that self-replicating loops can be used to solve the NP-complete problem known as satisfiability (SAT). Given a Boolean predicate like $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$, the problem is to find the assignment of Boolean values to the binary variables x_1, x_2, and x_3 that satisfies the predicate, that is, makes it evaluate to True (if such an assignment exists). In the above works of Tempesti and Perrier et al., the program embedded in each loop is copied unchanged from parent to child so that all replicated loops carry out the same program. Chou and Reggia took a different approach in which each replicant receives a distinct partial solution that is modified during replication. Under a form of artificial selection, replicants with promising solutions proliferate while those with failed solutions are lost. The process is demonstrated in Figure 10. This work shows how a cellular automaton can be used as a truly massively parallel machine to solve a difficult problem. This is in contrast to many works (including von Neumann's seminal one) where the highly parallel CA is used in a completely serial manner (e.g., by embedding a sequential Turing machine). As noted by Burks [84] (page 157): "Thus von Neumann's cellular structure allows for an indefinite amount of parallelism. But in designing his self-reproducing automaton von Neumann did not make much use of the potential parallelism of his cellular structure. Rather, his self-reproducing automaton works like a serial digital computer. . . ." Chou and Reggia observed that their system can be considered a form of DNA comput-

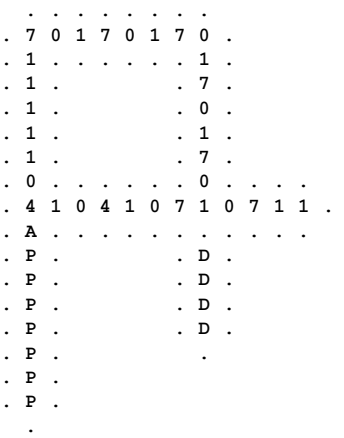


Figure 9. A self-replicating loop with programmable capabilities [61]. The system consists of three parts—loop, program, and data—all of which are replicated, followed by the program's execution on the given data. The cellular space is two-dimensional, five-neighbor, with 63 states per cell. P denotes a state belonging to the set of program states, D denotes a state belonging to the set of data states. Note that though the number of states seems prohibitive (63), the vast majority of entries in the rule table are identity transformations (i.e., ones that do not change the state of the central cell). This renders the automaton completely realizable (the transition rule is actually much simpler than that of von Neumann's even though it involves more states per cell).

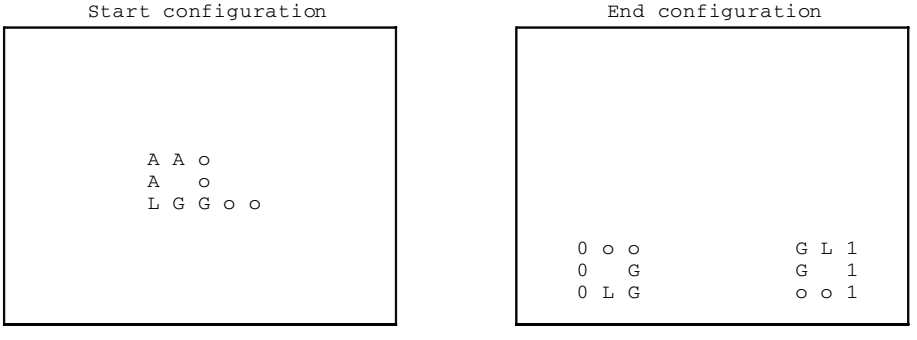


Figure 10. Solving the satisfiability (SAT) problem with self-replicating loops [15]. Shown above for a three-variable problem: $(\neg x_1 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$. The initial configuration of the cellular automaton contains a single loop with three embedded binary bits (marked by A's). This loop self-replicates in the cellular space, with each daughter loop differing by one bit from the parent, thus resulting in a parallel enumeration process. This is coupled with artificial selection that culls unfit solutions, by eliminating the loops that represent them (each loop represents one possible SAT solution). In the end only two loops remain, containing the two truth assignments for the predicate in question: $x_1, x_2, x_3 = 0, 0, 0$ or 1, 1, 1.

ing [1] in a cellular automaton using self-replicating loops in a vastly parallel fashion. Furthermore, molecular implementations can be imagined, for example, using synthetic self-replicators [66].

2.5 Modifying the Underlying CA Model

In this last subsection on CA-based works I describe a number of approaches in which the underlying model is no longer that of the classical, Ulam–von Neumann CA. Arbib [2, 3] presented a universal array in which the self-replicating structure is simpler to program (with respect to von Neumann's system) yet built of more complex elemental

cells. The basic unit, or cell, is a finite automaton that can execute an internal program of up to 20 instructions. Arbib [2] noted that von Neumann had

> ... shown that one may construct self-reproducing universal arrays using as basic cells finite automata with only 29 states. The price we pay for the simplicity of the components is that the coding of the array is enormously complicated, and the operation of the array requires many many steps to simulate one cycle of an ordinary Turing machine. (p. 179)

With respect to his model he noted that

> The price we pay for the simplicity of programming and operation is that our cells are more complicated.... The point of our construction is not that very simple or very complex components can be used to build a self-reproducing automaton; but rather that, given components of one level of complexity, we may use them to obtain self-reproducing aggregates of an arbitrarily higher level of complexity.... (p. 179)

Mange and his colleagues have been developing the Embryonics (Embryonic Electronics) project since 1993, whose ultimate objective is the construction of large-scale integrated circuits, exhibiting properties such as self-repair (healing), self-replication, and evolution, found until now only in living beings [48–50]. Such systems will be more robust than current-day ones, able to function within complex dynamic environments, which not only cannot be fully specified in advance, but furthermore may change in time. Essentially, Embryonics is a CA-based approach in which three biologically inspired principles are employed: multicellular organization, cellular differentiation, and cellular division.

The Embryonics team developed an artificial cell, dubbed *biodule* (biological module), that is used as an elementary unit from which multicellular organisms can ontogenetically develop [68, 75] to perform useful tasks. One of these organisms—the "biowatch"—is shown in Figure 11. Cellular differentiation takes place by having each cell compute its coordinates (i.e., position) within a one- or two-dimensional space, after which it can extract the specific gene within the artificial genome responsible for the cell's functionality (each cell contains the entire genome). Cellular division occurs when a mother cell, the *zygote*, arbitrarily placed within the grid, multiplies to fill a large portion of the space, thus forming a multicellular organism. In addition to self-replication, this artificial organism also exhibits self-repair capabilities, another biologically inspired phenomenon, lacking in the systems presented up until now. Mange et al. observed that such self-replicating machines are *multicellular* artificial organisms, in the sense that each of the several cells composing the organism contains one copy of the complete genome. In this respect, most other self-replicating automata described herein can be considered *unicellular* organisms: There is a single genome describing (and contained within) the entire machine (e.g., the Langton-like and von Neumann-like loops, described above).

Sipper [71–73] devised a small, five-cell, self-replicating loop, embedded in a two-dimensional, nine-neighbor, three-state cellular space. The underlying model is that of a *nonuniform* cellular automaton, in which the local update rule need not be identical for all grid cells (as is the case with the classical CA model). Furthermore, the cells are somewhat more complex than those of the original CA: Whereas a cell in the original model accesses the states of its neighbors but may change only its own state, Sipper's

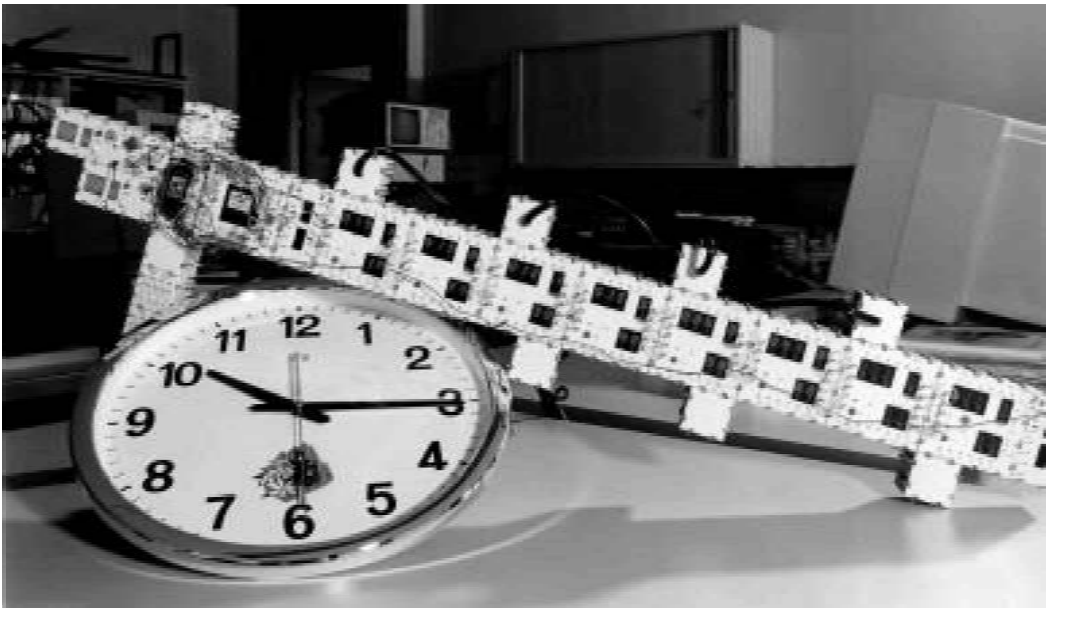


Figure 11. Embryonics project: the biowatch. An artificial "organism" designed to count minutes (from 00 to 59) and seconds (from 00 to 59), composing in effect a modulo-3600 counter that is able to self-replicate and self-repair. It is implemented using eight biodule cells, the basic unit of the Embryonics project.

model allows state changes of neighboring cells and rule copying into them (this latter characteristic can be considered a form of cellular movement).

Lohn and Reggia's CA model [45–47], used in their evolutionary experiments (Section 2.3), is also different from the classical CA. Similar to Sipper, they considered movable automata, called effector automata, embedded in a cellular space. Furthermore, they introduced a new method of automata input, called orientation-insensitive input, which was shown to increase the yield of self-replicating structures found by evolution.

3 Self-Replicating Programs

As we saw in Section 2.5, Arbib [2, 3]—while remaining within the cellular framework—considered the use of more complex cells, composing an internal program of up to 20 instructions. Departing altogether from the cellular model, Bratley and Millo [7] and Burger, Brill, and Machi [8] devised self-replicating computer programs. Core War, a computer game initiated by Dewdney in the 1980s, gained wide attention at the time [18–21]. Basically, it involves a virtual computer environment in which computer programs "do battle" with each other, the object being to destroy other programs and to occupy more virtual territory. Dewdney, then a columnist with *Scientific American*, invited readers to submit programs, conducting tournaments in his computer. Some of the Core War programs were able to self-replicate.

The question of whether open-ended evolution can be embedded within a computer was posed by Ray, who devised a virtual world called *Tierra*, consisting of computer programs that can undergo evolution [64, 65]. In contrast to evolutionary computation techniques where fitness is defined by the user [23, 52], the Tierra "creatures" (programs) receive no such direction. Rather, they compete for the natural resources of their computerized environment, namely, CPU time and memory. Because only a

finite amount of these are available, the virtual world's natural resources are limited, as in nature, serving as the basis for competition between creatures. Ray inoculated his system with a single, self-replicating organism, called the "Ancestor," which is the only engineered (manmade) creature in Tierra. He then set his system loose and witnessed the emergence of an ecosystem within the Tierra world, including organisms of various sizes, parasites, hyperparasites, and so on. The evolved parasites, for example, are small creatures that use the replication code of larger organisms (such as the Ancestor) to self-replicate. In this manner they proliferate rapidly without the need for the excess replication code. Being an ecologist, Ray was particularly interested in the emergence of such diverse ecological communities, using them to examine experimentally ecological and evolutionary processes, such as competitive exclusion and coexistence, host/parasite density-dependent population regulation, the effect of parasites in enhancing community diversity, evolutionary arms races, punctuated equilibria, and the role of chance and historical factors in evolution. He concluded that such "evolution in a bottle" may prove to be a valuable tool for the study of evolution and ecology. Ray has recently extended his Tierra environment to run on the Internet, rather than on a single computer, hoping that by increasing the scale of the system, new phenomena may arise that have not been observed on a single computer.

Working within the genetic programming framework [37], which essentially involves the evolution of Lisp programs, Koza [38] reported on the spontaneous emergence of computer programs. These exhibited the ability to reproduce asexually, as well as to reproduce by combining parts from two parents. Generating 12,500,000 programs at random, composed of a small number of elemental units (known as *functions* and *terminals* in Lisp jargon), Koza [38] discovered that a few of them were self-replicators. Thus, he showed that "spontaneous emergence of self-reproducing computer programs is possible" (p. 244). Koza [38] concluded, "The fact that spontaneous emergence can occur with a probability of the order of 10^{-6} to 10^{-9} with a function set such as $\mathcal{F}$ suggests that many fruitful experiments on spontaneous emergence and evolutionary self-improvement can be conducted at this time" (p. 259).

4 Self-Replicating Strings

In this section we shall describe a number of works with stringlike elements as the basic component. The artificial molecular machines introduced by Laing [39–41] are chains (possibly folded and interconnected) of basic moleculelike constituents. As put by Laing [41]:

> The basic components of our system consist of strands or *strings* of primitive constituent finite state automata, these component strings being in sliding contact.... A primitive constituent of a string can be in an activated or passive state. An active primitive constituent in contact with a passive constituent of another string will interact with the passive constituent in precisely defined ways only. These ways include *changing* the state of the contacted passive primitive, *reacting* to the state of the contacted passive primitive, *sliding* to the next neighbor of the contacted passive primitive. Since one string (an active string) can be designed to play the part of any Turing machine finite-state read-head, and another string (a passive string) can be designed to play the part of a Turing machine tape, we can carry out any Turing computation in this kinematic machine system.

Figure 12 shows a Turing machine in this format.

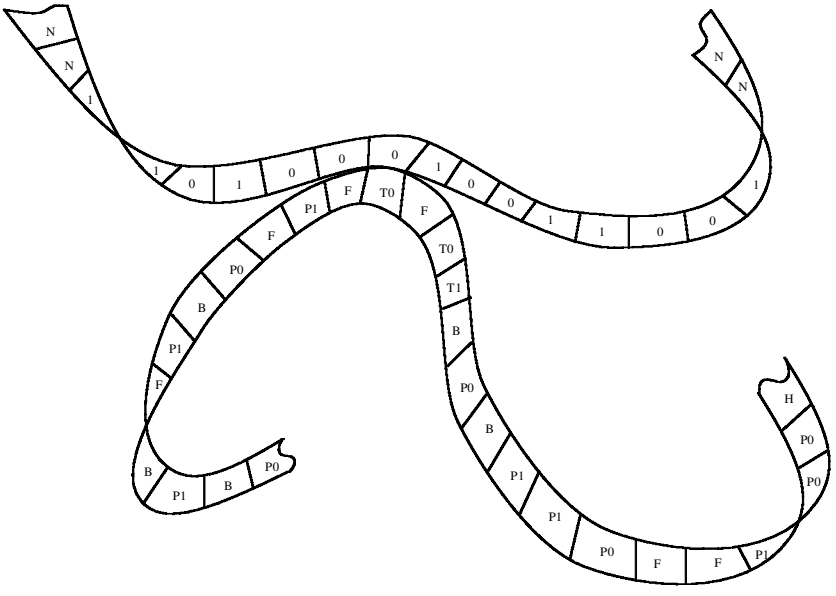


Figure 12. A Turing machine in Laing's kinematic machine format [41]. The lower string consists of program primitives that if activated will act on the passive "tape" primitives of which the upper string is composed. There are three primitives that are mainly employed in the passive tape: N (*null*), 0 (*zero*), and 1 (*one*). The program primitives include: P0 (*print zero*), P1 (*print one*), F (*forward*), B (*backward*), H (*halt*), TN, T0, and T1 (*conditional transfer* primitives), A (*activation*), AD (*activate and detach*), and C (*conversion*).

Replication in Laing's model is achieved by *self-inspection*, where the description of the object to be replicated (the genome) is dynamically constructed concomitantly with its interpretation. This is different from the other systems described herein (except that of Ibánez et al. [34]) where the genome is essentially predetermined (either by direct design or by artificial evolution). Laing [41] noted, "The capacity of a system generally to explore its own structure and produce a complete description of it for its perusal and use (for example, in generation and evaluation of behavioral options open to it) seems a valuable one, and if such a *prima facie* advantageous capacity is *not* exhibited anywhere in naturally occurring systems, this in itself seems of interest" (p. 455). As noted in Section 2.3, the self-replicating loops devised by Ibánez et al. [34], though implemented in a cellular automaton space, are also based on self-inspection methods.

Morris [59] studied self-replication within the typogenetics framework. Typogenetics was first introduced by Hofstadter [32] as a formal system for describing operations on DNA strands. A typogenetics string, or *strand*, has a double aspect: It is a coded message prescribing operations, and it is the very operand or data those operations will work on. Self-replication in typogenetics can be achieved in two ways [59]: (a) A string can extend itself horizontally along one level and then cut itself into two pieces that are either already replicas of their parent or will beget such replicas, or (b) it can use a copy operation to create a double strand that will separate into two daughters that are either already copies of their parent or will grow into such copies. Varetto [79] also studied the self-replication issue using the typogenetics model.

5 Other Works

As noted in Section 1, I have concentrated in this article on computational models, that is, ones that have been studied from a computer science point of view, be it theoretical or experimental. In this section I make a brief note of some other approaches.

In 1959 Lionel Penrose, aided by his son Roger, built simple mechanical units or bricks, an ensemble of which were placed in a box, which was then shaken. As described by Penrose [60]:

> In fanciful terms, we visualized the process of mechanical self-replication proceeding somewhat as follows: Suppose we have a sack or some other container full of units jostling one another as the sack is shaken and distorted in all manner of ways. In spite of this, the units remain detached from one another. Then we put into the sack a prearranged connected structure made from units exactly similar to those already within the sack.... Now we agitate the sack again in the same random and vigorous manner, with the seed structure jostling about among the neutral units. This time we find that replicas of the seed structure have been assembled from the formerly neutral or "lifeless" material. (p. 106)

Other simple mechanical models are those of Jacobson [35], who built a replicator using toy train parts running around a track, and Morowitz [58], who also constructed a simple physical replicator. Finally, two other early works are those of Kemeny [36], who discussed, among others, von Neumann's model, and Moore [54], who speculated on applications for machines that can reproduce.

Freitas, Jr. [24] presented the first quantitative engineering analysis of a complete self-replicating interstellar probe, with special attention to material, structural, and functional *closure* issues, referring to the possibility of finding the materials necessary for replication in the immediate environment (e.g., on the moon, in case of a lunar replicator). Some far-future space applications of machine replication technology were examined in two subsequent works [26, 30]. In 1980 NASA convened a committee of experts to conduct an in-depth study of various issues related to space exploration. One of the issues studied was the possibility of planting a "seed" factory on the moon that would then self-replicate to populate a large surface, using local lunar material [25, 27–29]. This study also introduced the concept of closure engineering, studying qualitative closure (can all parts be made?), quantitative closure (can enough parts be made?), and throughput closure (can parts be made fast enough?). (Apart from Laing, whose work was described in Section 4, and Freitas, Jr., both of whom participated in the NASA study, von Tiesenhausen and Darbro's work [85] also served as a precursor.)

Finally, it is worth mentioning the recent efforts in creating self-replicating molecules, one example of which is the work of Rebek, Jr. [66]:

> Imagine a molecule that likes its own shape: finding a copy of itself, it will fit neatly with its twin, forming for a while a complete entity. If the original molecule is presented with the component parts of itself, it will assemble these into additional replicas. The process will continue as long as the supply of components lasts. My colleagues and I ... have designed such self-assembling molecules and crafted them in the laboratory.... Our organic molecules, although they operate outside of living systems, help to elucidate some of the essential principles of self-replication. (p. 48)

It is interesting to note the *prima facie* similarity between this system and that of Penrose.

6 Concluding Remarks

My aim herein has been to present an overview of the major research efforts carried out over the past 50 years in the area of self-replication. As we approach the new millennium, it seems that there is a rekindling of interest in this issue. This partly stems from the appearance on the scene of new materials, such as synthetic molecules and nanomachines, that may ultimately be used to implement self-replicating structures on a very small scale.

While these past decades have been characterized by VLSI technology (very large-scale integrated circuits), perhaps the next few decades will see the coming of VSRM—very small-scale replicating machines.

Acknowledgments

I am grateful to the following people who provided comments that were helpful in the writing of this article: John Case, Robert A. Freitas, Jr., Chris Langton, Jason Lohn, Marco Tomassini, and Paul M. B. Vitányi. This work was supported in part by Grant 2000-049349.96 from the Swiss National Science Foundation.

References

1. Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science, 266*, 1021–1024.
2. Arbib, M. A. (1966). Simple self-reproducing universal automata. *Information and Control, 9*, 177–189.
3. Arbib, M. A. (1969). *Theories of abstract automata*. Englewood Cliffs, NJ: Prentice-Hall.
4. Bennett, C. H. (1988). Notes on the history of reversible computation. *IBM Journal of Research and Development, 32*(1), 16–23.
5. Bennett, C. H., & Landauer, R. (1985, July). The fundamental limits of computation. *Scientific American*, 48–56.
6. Beuchat, J.-L., & Haenni, J.-O. (1998). Von Neumann's 29-state cellular automaton: A hardware implementation. Manuscript submitted for publication.
7. Bratley, P., & Millo, J. (1972). Computer recreations: Self-reproducing programs. *Software Practice and Experience, 2*, 397–400.
8. Burger, J., Brill, D., & Machi, F. (1980). Self-reproducing programs. *Byte, 5*, 72–74.
9. Burks, A. (Ed.). (1970). *Essays on cellular automata*. Urbana: University of Illinois Press.
10. Byl, J. (1989). Self-reproduction in small cellular automata. *Physica D, 34*, 295–299.
11. Case, J. (1971). A note on degrees of self-describing Turing machines. *Journal of the Association for Computing Machinery, 18*, 329–338.
12. Case, J. (1974). Periodicity in generations of automata. *Mathematical Systems Theory, 8*, 15–32.
13. Case, J. (1974). Recursion theorems and automata which construct. In *Proceedings of the 1974 IEEE Conference on Biologically Motivated Automata Theory*. New York: IEEE.
14. Chou, H.-H., & Reggia, J. A. (1997). Emergence of self-replicating structures in a cellular automata space. *Physica D, 110*(3–4), 252–276.
15. Chou, H.-H., & Reggia, J. A. (1998). Problem solving during artificial selection of self-replicating loops. *Physica D, 115*(3–4), 293–372.

16. Codd, E. F. (1968). *Cellular automata*. New York: Academic Press.

17. Devore, J., & Hightower, R. (1992). The Devore variation of the Codd self-replicating computer. Original work carried out in the early 1970s though apparently never published. Presented at the Third Workshop on Artificial Life, Santa Fe, NM.

18. Dewdney, A. K. (1984, May). Core war. *Scientific American, 250*(5), 15–19.

19. Dewdney, A. K. (1985, March). Core war. *Scientific American, 252*(3), 14–19.

20. Dewdney, A. K. (1987, January). Core war tournament. *Scientific American, 256*(1), 8–11.

21. Dewdney, A. K. (1989, March). Of worms, viruses and core war. *Scientific American, 260*(3), 90–93.

22. Drexler, K. E. (1992). *Nanosystems: Molecular machinery, manufacturing and computation*. New York: John Wiley.

23. Fogel, D. B. (1995). *Evolutionary computation: Toward a new philosophy of machine intelligence*. Piscataway, NJ: IEEE Press.

24. Freitas, R. A., Jr. (1980, July). A self-reproducing interstellar probe. *Journal of the British Interplanetary Society, 33*, 251–264.

25. Freitas, R. A., Jr. (1981, September). Report on the NASA/ASEE summer study on advanced automation for space missions. *Journal of the British Interplanetary Society, 34*, 407–408.

26. Freitas, R. A., Jr. (1983, March). Terraforming Mars and Venus using machine self-replicating systems. *Journal of the British Interplanetary Society, 36*, 139–142.

27. Freitas, R. A., Jr., & Gilbreath, W. P. (Eds.). (1982). *Advanced automation for space missions: Proceedings of the 1980 NASA/ASEE summer study*, chapter 5: Replicating Systems Concepts: Self-replicating Lunar Factory and Demonstration. NASA, Scientific and Technical Information Branch (Conference Publication 2255), Washington, DC: U.S. Government Printing Office.

28. Freitas, R. A., Jr., Healy, T. J., & Long, J. E. (1981). Advanced automation for space missions. In P. J. Hayes (Ed.), *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI-81)* (pp. 803–808). Los Altos, CA: William Kaufmann.

29. Freitas, R. A., Jr., Healy, T. J., & Long, J. E. (1982, January–March). Advanced automation for space missions. *Journal of the Astronautical Sciences, 30*, 1–11.

30. Freitas, R. A., Jr., & Valdes, F. (1980, November). Comparison of reproducing and non-reproducing starprobe strategies for galactic exploration. *Journal of the British Interplanetary Society, 33*, 402–408.

31. Garzon, M. (1995). *Models of massive parallelism: Analysis of cellular automata and neural networks*. Berlin: Springer-Verlag.

32. Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An eternal golden braid*. Hassocks, UK: The Harvester Press.

33. Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory languages and computation*. Redwood City, CA: Addison-Wesley.

34. Ibánez, J., Anabitarte, D., Azpeitia, I., Barrera, O., Barrutieta, A., Blanco, H., & Echarte, F. (1995). Self-inspection based reproduction in cellular automata. In F. Morán, A. Moreno, J. J. Merelo, & P. Chacón (Eds.), *ECAL'95: Third European Conference on Artificial Life* (vol. 929 of *Lecture Notes in Computer Science*, pp. 564–576). Heidelberg: Springer-Verlag.

35. Jacobson, H. (1958). On models of reproduction. *American Scientist, 46*, 255–284.

36. Kemeny, J. G. (1955, April). Man viewed as a machine. *Scientific American, 192*, 58–68.

37. Koza, J. R. (1992). *Genetic programming*. Cambridge, MA: MIT Press.

38. Koza, J. R. (1994). Artificial life: Spontaneous emergence of self-replicating and evolutionary self-improving computer programs. In C. G. Langton (Ed.), *Artificial life III* (pp. 225–262). Reading, MA: Addison-Wesley.

39. Laing, R. (1975). Some alternative reproductive strategies in artificial molecular machines. *Journal of Theoretical Biology, 54*, 63–84.

40. Laing, R. (1976). Automaton introspection. *Journal of Computer and System Sciences, 13*, 172–183.

41. Laing, R. (1977). Automaton models of reproduction by self-inspection. *Journal of Theoretical Biology, 66*, 437–456.

42. Langton, C. G. (1984). Self-reproduction in cellular automata. *Physica D, 10*, 135–144.

43. Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D, 22*, 120–149.

44. Lindenmayer, A. (1968). Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology, 18*, 280–315.

45. Lohn, J. D. (1996). *Automated discovery of self-replicating structures in cellular space automata models*. (Tech. Rep. CS-TR-3677). Department of Computer Science, University of Maryland at College Park.

46. Lohn, J. D., & Reggia, J. A. (1995). Discovery of self-replicating structures using a genetic algorithm. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation (ICEC'95)* (pp. 678–683). Piscataway, NJ: IEEE.

47. Lohn, J. D., & Reggia, J. A. (1997). Automatic discovery of self-replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation, 1*(3), 165–178.

48. Mange, D., Goeke, M., Madon, D., Stauffer, A., Tempesti, G., & Durand, S. (1996). Embryonics: A new family of coarse-grained field-programmable gate array with self-repair and self-reproducing properties. In E. Sanchez & M. Tomassini (Eds.), *Towards evolvable hardware* (vol. 1062 of *Lecture Notes in Computer Science*, pp. 197–220). Heidelberg: Springer-Verlag.

49. Mange, D., Madon, D., Stauffer, A., & Tempesti, G. (1997). Von Neumann revisited: A Turing machine with self-repair and self-reproduction properties. *Robotics and Autonomous Systems, 22*(1), 35–58.

50. Mange, D., & Tomassini, M. (Eds.). (1998). *Bio-inspired computing machines: Toward novel computational architectures*. Lausanne, Switzerland: Presses Polytechniques et Universitaires Romandes.

51. Merkle, R. C. (1992). Self-replicating systems and molecular manufacturing. *Journal of the British Interplanetary Society, 45*, 407–413.

52. Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd. ed.). Heidelberg: Springer-Verlag.

53. Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

54. Moore, E. F. (1956, October). Artificial living plants. *Scientific American, 195*, 118–126.

55. Morita, K., & Imai, K. (1996). Self-reproduction in a reversible cellular space. *Theoretical Computer Science, 168*, 337–366.

56. Morita, K., & Imai, K. (1997). Logical universality and self-reproduction in reversible cellular automata. In T. Higuchi, M. Iwata, & W. Liu (Eds.), *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)* (vol. 1259 of *Lecture Notes in Computer Science*, pp. 152–166). Heidelberg: Springer-Verlag.

57. Morita, K., & Imai, K. (1997). A simple self-reproducing cellular automaton with shape-encoding mechanism. In C. Langton & T. Shimohara (Eds.), *Artificial life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*. Cambridge, MA: MIT Press.

58. Morowitz, H. J. (1959). A model of reproduction. *American Scientist, 47*, 261–263.

59. Morris, H. C. (1989). Typogenetics: A logic for artificial life. In C. G. Langton (Ed.), *Artificial life* (pp. 369–395). Redwood City, CA: Addison-Wesley.

60. Penrose, L. S. (1959). Self-reproducing machines. *Scientific American, 200*(6), 105–114.

61. Perrier, J.-Y., Sipper, M., & Zahnd, J. (1996). Toward a viable, self-reproducing universal computer. *Physica D, 97*, 335–352.

62. Pesavento, U. (1995). An implementation of von Neumann's self-reproducing machine. *Artificial Life, 2*(4), 337–354.

63. Prusinkiewicz, P., & Lindenmayer, A. (1990). *The algorithmic beauty of plants*. New York: Springer-Verlag.

64. Ray, T. S. (1992). An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 371–408). Redwood City, CA: Addison-Wesley.

65. Ray, T. S. (1994). An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life, 1*, 179–209.

66. Rebek, J., Jr. (1994, July). Synthetic self-replicating molecules. *Scientific American, 271*(1), 48–55.

67. Reggia, J. A., Armentrout, S. L., Chou, H.-H., & Peng, Y. (1993, February). Simple systems that exhibit self-directed replication. *Science, 259*, 1282–1287.

68. Sanchez, E., Mange, D., Sipper, M., Tomassini, M., Pérez-Uribe, A., & Stauffer, A. (1997). Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In T. Higuchi, M. Iwata, & W. Liu (Eds.), *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)* (vol. 1259 of *Lecture Notes in Computer Science*, pp. 35–54). Heidelberg: Springer-Verlag.

69. Signorini, J. (1989). How a SIMD machine can implement a complex cellular automaton? A case study: Von Neumann's 29-state cellular automaton. In *Supercomputing '89: Proceedings of the ACM/IEEE Conference* (pp. 175–186). New York: ACM Press.

70. Signorini, J. (1990). Complex computing with cellular automata. In P. Manneville, N. Boccara, G. Y. Vichniac, & R. Bidaux (Eds.), *Cellular automata and modeling of complex physical systems* (vol. 46 of *Springer Proceedings in Physics*, pp. 57–72). Heidelberg: Springer-Verlag.

71. Sipper, M. (1994). Non-uniform cellular automata: Evolution in rule space and formation of complex structures. In R. A. Brooks & P. Maes (Eds.), *Artificial life IV* (pp. 394–399). Cambridge, MA: MIT Press.

72. Sipper, M. (1995). Studying artificial life using a simple, general cellular model. *Artificial Life, 2*, 1–35.

73. Sipper, M. (1997). *Evolution of parallel cellular machines: The cellular programming approach*. Heidelberg: Springer-Verlag.

74. Sipper, M., Mange, D., & Stauffer, A. (1997). Ontogenetic hardware. *BioSystems, 44*(3), 193–207.

75. Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Pérez-Uribe, A., & Stauffer, A. (1997). A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation, 1*(1), 83–97.

76. Smith, A. R. (1992). Simple nontrivial self-reproducing machines. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II* (pp. 709–725). Redwood City, CA: Addison-Wesley. (Originally part of Smith's Ph.D. dissertation: *Cellular Automata Theory*, Tech. Rep. No. 2, Digital Systems Laboratory, Stanford University, Stanford, CA, 1969).

77. Stauffer, A., & Sipper, M. (1998). On the relationship between cellular automata and L-systems: The self-replication case. *Physica D, 116*(1–2), 71–80.

78. Tempesti, G. (1995). A new self-reproducing cellular automaton capable of construction and computation. In F. Morán, A. Moreno, J. J. Merelo, & P. Chacón (Eds.), *ECAL'95: Third European Conference on Artificial Life* (vol. 929 of *Lecture Notes in Computer Science*, pp. 555–563). Heidelberg: Springer-Verlag.

79. Varetto, L. (1993). Typogenetics: An artificial genetic system. *Journal of Theoretical Biology, 160*, 185–205.

80. Villasenor, J., & Mangione-Smith, W. H. (1997, June). Configurable computing. *Scientific American, 276*(6), 54–59.

81. Vitányi, P. M. B. (1973). Sexually reproducing cellular automata. *Mathematical Biosciences, 18*, 23–54.

82. Vitányi, P. M. B. (1974). Genetics of reproducing automata. In *Proceedings of the 1974 IEEE Conference on Biologically Motivated Automata Theory* (pp. 166–171). New York: IEEE.

83. Vitányi, P. M. B. (1976). On a problem in the collective behavior of automata. *Discrete Mathematics, 14*, 99–101.

84. von Neumann, J. (1966). *Theory of self-reproducing automata*. Urbana: University of Illinois Press. Edited and completed by A. W. Burks.

85. von Tiesenhausen, G., & Darbro, W. A. (1980). Self-replicating systems—a systems engineering approach. Tech. Memo. TM-78304. NASA.

86. Wolfram, S. (1994). *Cellular automata and complexity*. Reading, MA: Addison-Wesley.