Introduction to and Survey of Cellular Automata or Polyautomata Theory[1]

Alvy Ray Smith III
Computer Graphics Laboratory
New York Institute of Technology
Old Westbury, N.Y. 11568, U.S.A.

Published in: **Automata, Languages, Development**, eds. A. Lindenmayer and G. Rozenberg, North-Holland Publishing Co., New York, 1976, 405-422, a book which serves as the proceedings of the conference Formal Languages, Automata and Development, Noordwijkerhout, the Netherlands, Apr 1975. I style this conference Artificial Life 0 since it brought together for the first time biological and computer scientists many of whom would later attend the conferences formally known as Artificial Life 1 (ALife1), Artificial Life 2 (ALife2), etc.

This document was reentered by Alvy Ray Smith in Microsoft Word form on May 21, 2001. Spelling and punctuation are generally preserved—except "neighbour" is replaced everywhere with "neighbor"—but trivially minor spelling errors are corrected. Underlining is replaced by italics in the text, by boldface in the bibliography, and square brackets replace parentheses in bibliographic references. Otherwise additions or changes made to the original are noted inside square brackets.

The cornerstone of polyautomata theory and the principal result established by this book[2] is that it is logically possible for a nontrivial computing machine to reproduce, or replicate, itself *ad infinitum*. The realization of this logical result in physical objects has not yet been accomplished, even on paper, except to the extent that a living, reproducing thing is a machine. Although this latter view has certainly been supported by the description of nuclear DNA programs in the living cell and the transcription of them into protein building blocks, it still takes a leap of faith to believe that von Neumann's result means it is physically possible for a general-purpose computer to reproduce itself. We have taken the chemicals of living things and made vital parts of living things from them (e.g., genes [62]), but we have not yet generalized the secrets of living things to non-living creations of our own. (By "non-living" I mean not a member of the biological kingdom with its history of evolution, man-made). I believe we shall, that the leap of faith is a small one and shall soon require no faith. I see forests of inorganic trees. I see buildings construct themselves, growing from a single brick-egg each. I see robots reproduce and evolve. I see amplification of our species and its celebration but also competition. There is an eerie beauty in these visions, generation-spanning mystery, the sense of frontier, but also the horrors possible from abuse,

---

[1] [Title changed by the addition of "Cellular Automata or" because "polyautomata" never became common currency. Actually, polyautomata are generalizations of CA as explained in this article.]

[2] The book referred to throughout this article is John von Neumann's **Theory of Self-Reproducing Automata**, edited by Arthur W. Burks, Univ. of Illinois Press, Urbana, 1968. The present article is an introduction to the German edition of this volume [which was never published].

shortsightedness, and habitual blindness. The following book is the down-to-earth origin of this visionary space, written by a founding father of computer science and pursued the seventeen years since his death by his followers in spirit, the computer scientists.

In this introduction, I shall survey and taxonomize the subbranch of computer science which I have chosen to call *polyautomata theory*, where a *polyautomaton* is a multitude of interconnected automata operating in parallel to form a larger automaton, a macroautomaton formed of microautomata. The mathematical device employed by von Neumann to solve the logical problems of self-reproduction—he sets aside his own attempt at physical realization early on—is a polyautomaton called the cellular space[3]. Thus this book is one of the first published documents of polyautomata theory, a branch of automata theory [9, 43]. As we shall see, particularly in the discussion of "dynamic" polyautomata, growth and development of living things, biological computations, are never far from the mind of the polyautomata theorist. Nor is the sense that the theory is leading him to a profounder understanding of parallel computation.

I have chosen to introduce the survey by chronicling the refinements of von Neumann's solution to the problem of self-reproduction. To understand his result it is necessary to establish some basic concepts of polyautomata theory. In particular, the cellular space is described in some detail; von Neumann's result is then presented, followed by its refinements. The developments of the theory of cellular spaces are catalogued, and then the generalization of cellular spaces to polyautomata is accomplished via a taxonomy which makes the statement of open questions straightforward. I believe the taxonomy clarifies a theory presently muddied by use of the same name for different polyautomata and different names for the same polyautomaton. I have selected the name "polyautomata theory" to be descriptive while avoiding problems of vested interest in any one of the many names used for members of the polyautomata family.

Not only is it appropriate to begin with the von Neumann cellular space for historical reasons, but it is also convenient, his model being rapidly grasped via our natural human inductive powers. For example, consider a chessboard extended to infinity in both dimensions with its rows and columns aligned with the north-south and east-west axes. Think of each square as representing an ordinary (serial) digital computer and suppose all these computers are identical. Furthermore, assume that each computer has input and output lines from and to its four nearest-neighbor cells—that is, the computers just to its north, south, east, and west. This hypothetical piece of hardware, extending to infinity in the four directions, is an intuitive model of the cellular space.

The abstract model utilized by polyautomata theorists, including von Neumann in this book, is easily obtained from the intuitive model by replacing each digital computer with an abstraction, an automaton, called a *finite-state machine*. Here "machine" means logical machine and does not necessarily imply anything mechanical. A finite-state machine is a theoretical device with a finite number of states, one and only one of which it must be in at any given time. For example,

[3] [This is the modern cellular automaton, or CA.]

the finite-state machine abstraction of a light switch has two states corresponding to the two stable positions of the switch. A *cell* is one of the infinity of finite-state machines in a cellular space (which is thus not a finite-state machine). The cell designed by von Neumann in this book is a 29-state finite-state machine. A much larger example is the finite-state machine abstraction of the Univac 1108 I used at New York University last year. It would have more than $2^{2^{22}}$ states since the computer has more than 2^{22} bits in its main memory. This is more than 10^{10^6} states, a one followed by a million zeroes! A cellular space with each cell of this size is a piece of theoretical hardware of staggering complexity, it may seem, but the concept of cellular space allows us to play at ease among these vast numbers.

Besides a finite number of states, a finite-state machine, and hence a cell, has only a finite number of distinct input values and only a finite number of distinct output values. In the von Neumann cellular space, the set of output values is the same as the set of states. Then, if a cell is in state *s*, its output (the value on the output lines in the intuitive model) is also *s*. It is distributed as input to the four nearest-neighbor cells and also fed back to the cell itself as input. These five cells form the *neighborhood* of each cell. The four cells in the neighborhood of a cell, excluding itself, are the *neighbors* of the cell. Hence the set of input values from each neighbor is also the same as the set of states.

To complete a description of a cell, it is only necessary to add a table that gives the next state of the cell, and hence its next output, given its present state and the present states of its neighbors, for all possible present state combinations. A large part of the following book is devoted to describing this table for a typical cell of one cellular space. It is quite large—29^5, or approximately 20 million, entries—but finite.

For a finite-state machine, time flows in the direction of present state to next state, the important points being the times of state transition. Hence a finite-state machine can be thought of as a clock that ticks at each change of state. Alternatively, a finite-state machine listens to a regularly ticking clock and changes state at each tick. It is this latter view that is most convenient for a cellular space. All cells are assumed to listen to the same clock. That is, they change state simultaneously.

There is a special state, called the *quiescent* state, for each cell which is the state all but a finite number of cells assume. (This finiteness condition is not always assumed [14, 54]). Intuitively, most cells are off, or *quiescent*, leaving a finite number on. A cell can turn on only if at least one of its neighbors is on and then only if the next-state table allows it. Thus, although a cellular space is infinite, only a finite portion of it is of interest at any given time. A *configuration* in a cellular space is the arrangement of states formed at any one time by every cell assuming one of its states. An *initial configuration* is the configuration existing at time zero—that is, the only time at which an external agent may act upon a cellular space, placing each cell in some starting state. All changes in configuration after time zero occur autonomously, with one change possible per time step.

The *support* of a configuration is the set of nonquiescent cells. By the finiteness condition mentioned above, configurations are usually assumed to have finite support. These are called *finite configurations* when it is desired to make the

finiteness explicit. Thus the support of a (finite) configuration at any one time corresponds intuitively to a parallel computer, a (possibly quite large) number of serial computers operating simultaneously and interdependently. The infinity of quiescent cells surrounding the support can be thought of as a soup, or sea, in which this parallel machine is afloat and from which it can extract hardware to append to its surfaces. This makes it appear to expand in the space. If it detaches hardware from its surfaces, by turning cells off, then it appears to contract.

It is just such a machine that is shown capable of self-reproduction in this book. It is a nontrivial machine because it is shown to be a general-purpose computer, capable of computing any computable function. The machine is specified, as the states of the cells in the support of an initial configuration, by an external agent, in this case von Neumann. The next-state table, also called the *local transition function*, then takes over at each cell. Since all cells change state simultaneously, this causes a sequence of configurations to unfold, one per tick of the clock. The sequence can be visualized as a cartoon of nonquiescent states against a background of quiescent states. The initial machine begins to move, by expansion and contraction. It extends an arm into quiescent regions of the space where it places new cells on a machine it constructs there (by turning the cells on and placing them in some appropriate nonquiescent state). When the machine it constructs is a copy of itself, then it is said to self-reproduce.

I stated in the first paragraph that the physical realization of a self-reproducing machine had not yet occurred. With the cellular space concepts developed so far, we can obtain a feeling for the distance separating the theoretical from the actual. It is straightforward to realize a finite-state machine with electronic components—any computer engineer can do it—though perhaps inefficiently. Hence any cell is realizable and any finite set of cells. It is not straightforward, however, to exchange the theoretically convenient infinite hardware soup for real non-homogeneous space surrounding an artificial mechanical organism—it with organs capable of acquiring raw materials from the space and converting them into new cells which the organism grows as and where needed. To the extent one can tolerate a homogeneous model of nonhomogeneous space and the discontinuity at the surfaces of the machine is the demonstration in this book a proof ot the existence of real-world self-reproducing machines.

What it does unequivocally prove though is that a very large array of not very powerful computers operating in parallel can be programmed to be quite powerful—that is, to perform any computation digital computers can compute—and that the program to accomplish this feat can spread copies of itself throughout the array. So, not only is the array of relatively powerless computers capable of computing what one powerful computer can, but it is capable of computing what a large number of powerful computers can compute simultaneously, with only one of the programs being provided by man.

The von Neumann result has been refined and simplified several times in several ways. In the nine years between von Neumann's work and the first publication of it by editor Burks, Thatcher, a former student of Burks, published a 50-page version of the proof using the 29-state cells of the von Neumann cellular space [97]. Codd [22] reduced the size of the cell to 8 states in a proof of 80 pages.

Arbib [7] reduced the length of the proof to 8 pages by using very large cells (2^{335} states). Most recently, Banks [14] has demonstrated in about 20 pages the existence of nontrivial self-reproducing machines in a cellular space with only 4 states per cell. In all these cases, the 5-cell von Neumann neighborhood is assumed, and nontriviality means *computation universal* in the sense that the two-dimensional self-reproducing machine can compute any computable functino. In addition, each of these self-reproducing machines can construct any of a large class of machines over which it is said to be a *universal constructor*. Of course, this class of machines includes the self-reproducing machine itself. I have shown [90] that one dimension suffices and that the concept of universal constructor is not necessary for the existence of computation-universal self-reproducing machines. This proof is accomplished, in about three pages, in a one-dimensional cellular space (one row of a two-dimensional cellular space) with 18-state cells and the 3-cell nearest-neighbor neighborhood[4]. It is also shown that one nearest neighbor suffices if 40-state cells are assumed and that 2-state cells suffice if the neighborhood is increased to 21 cells. This is an example of the state-neighborhood size trade-off, to be mentioned again, possible in cellular spaces.

If the condition that a self-reproducing machine be nontrivial is relaxed, then self-reproduction is obtained trivially: Consider a one-dimensional cellular space with 2-state cells (0 and 1) and a 2-cell neighborhood consisting of a cell and its left nearest neighbor. Let 0 be the quiescent state, and let a single cell in state 1 be the only nonquiescent cell in the initial configuration. Then this single cell is a one-cell self-reproducing machine under local transition function f given by $f(01) = f(10) = f(11) = 1$ and $f(00) = 0$. More interestingly, there are cellular spaces [4, 63, 72, 89, 105] in which the *pattern*—that portion of a finite configuration restricted to its support—of every initial configuration self-reproduces.

The existence of nontrivial self-reproducing machines is only one of a "classic" set of properties a cellular space might possess. Given a cellular space one may ask if it is *construction universal* over some class of machines. This class is usually taken to include a set of computers of the computable, or partial recursive, functions, where each computer has a representation in the cellular space. If a cellular space supports a universal constructor over the given class of machine, then it is clearly construction universal. It is not known whether construction universality implies the existence of a universal constructor. Analogously, a cellular space is computation universal if it supports a *universal computer*, one machine capable of being programmed to compute any computable function. Arbib [9] has shown that, with the usual assumption for computations of the partial recursive functions—namely that the "program" can always be separated from its "data", with perhaps a decoding—computation universality implies the existence of a universal computer. The von Neumann cellular space is construction

[4] [The small neighbor or state counts apply only to computation-universal CA, not necessarily to self-reproducing CA. Details of this correction are in Smith, Alvy Ray, Simple nontrivial self-reproducing machines, Artifical Life 2, Santa Fe Institute Studies in the Sciences of Complexity, vol X, ed by C G Langton, C Taylor, J D Farmer, and S Rasmussen, Addison-Wesley, 1991, 709-725. The brevity of the proof of nontrivial self-reproducing CA remains unchanged, however. This paper should also be consulted for further up-to-date results on topics covered here.]

and computation universal because it supports a universal constructor that is also a universal computer. The existence of a universal constructor that is a universal computer does not necessarily imply nontrivial self-reproduction unless the universal constructor is a member of the class of machines over which it is universal. This, however, has usually been the case as, for example, in this book. For a discussion of the definitions of universal computers and constructors, see Herman [37].

By the usual choice of the class of machines over which a construction-universal cellular space is universal, construction universality implies computation universality. It is not known whether computation universality implies construction univerality in general although this is the case for the one-dimensional cellular spaces in [92]. Similarly, it is not known whether computation universality implies nontrivial self-reproducing machines in general although this too is the case for the one-dimensional cellular spaces just noted.

The simplest known construction-universal cellular space is that with 4-state cells already mentioned in the discussion of self-reproduction. Its inventor, Banks, has also discovered the simplest known two-dimensional computation-universal cellular space with the 5-cell von Neumann neighborhood. It has 3-state cells [14]. From a theorem of Codd [22], it is the simplest possible computation-universal cellular space with this neighborhood—that is, there is no 2-state cell version possible. However, if the neighborhood is increased to include the four nearest diagonal neighbors as well as the four nearest orthogonal neighbors, then computation universality is possible with 2-state cells. This 9-cell neighborhood is called the Moore neighborhood after Moore [64]. The cellular space called the "game of 'life'", invented by J.H. Conway of Cambridge University [29, 30], has the Moore neighborhood and 2-state cells. It has been informally proved computation universal by Conway and independently by W. Gosper of the Massachusetts Institute of Technology. Their formal proofs would be very long and tedious; hence they will probably never write them down. A local, hence brief, test for computation universality is not known—i.e., one which could be applied to the local transition function of a cellular space. I have established the simplest known one-dimensional computation-universal cellular spaces [92] in the following "sizes", where the first number in each pair is the state-set size and the second is the neighborhood size: 2x21, 3x13, 4x9, 5x8, 7x6, 11x4, 18x3, 40x2. [5]

Another "classic" property of some cellular spaces is a *Garden-of-Eden* configuration. This is a configuration that cannot arise by autonomous operation of a cellular space but must be the initial configuration if it is to exist at all. The lack of existence of Garden-of-Eden configurations was originally thought to be an important indicator of construction universality—an unattainable configuration is not constructible. But construction is usually defined to be of a specific class of configurations (which compute), hence none of these configurations need be Garden-of-Eden even if the cellular space has Garden-of-Eden configurations. Furthermore, subtleties in the definition of construction make it possible for

[5] [Deleted: "All of these support nontrivial self-reproduction."]

some Garden-of-Eden configurations to be constructed [97]. Thus construction universality is possible in cellular spaces with Garden-of-Eden configurations—e.g., the von Neumann cellular space. It turns out, in fact, that very weak cellular spaces have Garden-of-Eden configurations—e.g., the cellular space defined above for trivial self-reproduction—and their existence is certainly not sufficient to guarantee computation universality [90]. Moore [64] proved that the existence of a pair of "mutually erasable" configurations implies the existence of a Garden-of-Eden configuration, where two configurations are mutually erasable, roughly speaking, if they differ at time t but become identical at time $t+1$ under action of the *global transition function*—i.e. simultaneous action of the local transition functions of all cells. For example, consider again the cellular space defined above for trivial self-reproduction. The two configurations 111 and 101 (assume 0 everywhere else) are both transformed by the global transition function into the configuration 1111. Hence 111 and 101 are mutually erasable, and there exists a Garden-of-Eden configuration—e.g. 1. Banks and his coworkers at the Massachusetts Institute of Technology have recently discovered an efficient method for deriving Garden-of-Eden configurations (unpublished daily log memo). One application of the technique resulted in a Garden-of-Eden configuration for the game of "life" [103].

Myhill [68] proved the converse—that existence of mutually erasable configurations is equivalent to existence of Garden-of-Eden configurations. Since a global transition function takes any configuration, the Moore and Myhill results can be stated in terms of properties of the function. Amoroso and Cooper [3], Richardson [78] and Golze [31] have made these properties precise: A one-to-one global transition function is both one-to-one and onto if it is restricted to finite configurations. The converse is not true; a restricted one-to-one global transition function (restricted to finite configurations) is not necessarily a restricted onto global transition function, and a restricted onto global transition function is not necessarily a one-to-one global transition function. However, a global transition function is onto if and only if its restriction to finite configurations is one-to-one. We shall return to the Garden-of-Eden problem after polyautomata more general than the cellular space are discussed.

Polyautomata theory has been pursued simultaneously along several fronts by independent researchers in the time since von Neumann and the "cellular space". A representative, but certainly not exhaustive, list includes Hennie [34] on "iterative circuits"; Waksman [104], Balzer [13], and Moore and Langdon [65] on the "firing squad problem"; Holland [40] on "iterative computers"; Wagner [101] on "modular computers"; Cole [23] and Fischer [28] on "iterative arrays"; Kosaraju [50] on "cellular arrays"; Moore [64] on "tessellation structures"; Yamada and Amoroso [107] on "tessellation automata"; Smith III [91] and Codd[22] on "cellular automata"; Kilmer [48] and Kasami and Fujii [47] on "iterative logic networks"; Herman [36] and Lindenmayer and Rozenberg [59] on "developmental systems"; Rosenstiehl, Fiksel, and Holliger [81] on "intelligent graphs"; Aladyev [1] on "homogeneous structures"; Case and Steward [21] on "local computer systems"; and Arnold, Tan, and Newborn [11] on "iterative realizations". The following taxonomy should help to standardize this plethora of names.

Every polyautomaton has these properties: It is an interconnection of cells. Each cell computes an output from inputs it receives from a finite set of cells, forming its *input neighborhood*, and possibly from an external source. Each cell computes an output at each tick of a clock, and the output is distributed to its *output neighborhood*, a finite set of cells, and possibly to an external receiver. Although it is usually assumed that the clock of a polyautomaton is a single global clock, ensuring that all cells compute synchronously, it has been shown that this is not always necessary for synchrony, that a local clock built into each cell suffices for some polyautomata [81].

The term "cell" is intended mean a finite-state machine, perhaps augmented with tapes or stacks, although little work has yet been done with polyautomata having cells so augmented (but see [79, 87]). Generalizations to probabalistic or fuzzy cells [9] are possible but will not be treated here. In this survey, it is assumed that all cells of a given polyautomaton are identical—the polyautomaton is *monogeneous*. This is not taken to be a defined restriction on polyautomata but a reflection of the fact that the branch of the theory allowing a variety of cell types (polygeneity) is unexplored except for the work of Holland [41] who has presented a careful definition of polyautomata in terms of "compositions of finite automata". Similarly, it is usually assumed here that any polyautomaton is *input-output symmetric*, that the input and output neighborhoods of a given cell are the same unless otherwise specified. It is also assumed here that input and output values are the states, no input encoding or output decoding being necessary. It is not assumed that a cell is necessarily in its own neighborhood. A final assumption about polyautomata local to this survey is that each cell has memory, or more than one state. That is, the polyautomata here are assumed to be not *combinational*.

The class of polyautomata can be divided into two (not necessarily disjoint) subclasses in several convenient ways. There are the *infinite* and *finite* polyautomata each being an interconnection of, respectively, an infinite number or a finite number of cells. These two subclasses are mutually exclusive, with the infinite polyautomata being more powerful, of course, than the finite. There are the *uniform* and the *nonuniform* polyautomata. A uniform polyautomaton has a uniform interconnection scheme, a "wiring diagram" that somehow "looks the same" viewed from any cell. This intuitive notion can be made rigorous in terms of group graphs [102]. The von Neumann cellular space, its interconnection scheme specified by the 5-cell von Neumann neighborhood, is an infinite uniform polyautomaton. A non-uniform polyautomaton has an interconnection scheme that is not necessarily uniform. The relationship between the uniform and nonuniform polyautomata is not completely understood. However, Jump and Kirtane [46] have recently determined some of the structure of this relationship by considering those nonuniform polyautomata, the *balanced* polyautomata, which are not necessarily input-output symmetric but have the size of the input neighborhood equal to the size of the output neighborhood. They found that the finite balanced polyautomata are of power equal to the finite uniform polyautomata but their technique does not generalize to the infinite case, the infinite balanced being perhaps more powerful than the infinite uniform polyautomata. As an example of a

finite nonuniform polyautomaton, consider a set of five cells, such that each has three neighbors, no two of which are the same cell, and at least one cell is not its own neighbor. The interconnection scheme of this example is specified by a finite bidirectional graph of degree 3. The cells can be visualized as occupying the nodes of this graph with the arcs specifying the neighborhood of each cell. They are bidirectional arcs in this example because input-output symmetry is assumed.

Another taxonomic distinction to be made is that between *static* and *dynamic* polyautomata. These terms can be defined by reference to the interconnection graph. If the cells and the neighborhoods of all cells in a polyautomaton remain unchanged in time, then it is a static polyautomaton. That is, if the interconnection graph remains unchanged in time, the polyautomaton is static. If the arcs can change in number and position but the nodes are fixed, then the correspponding polyautomaton is only *node-static* but still considered static. A dynamic polyautomaton is not static. Both the arcs and nodes can change in number and position in its interconnection graph. Thus the dynamic polyautomata are those to which cells can be added in time and from which cells can be deleted. Except for the random pairwise interconnections considered in [100], node-static polyautomata have received scant attention. The dynamic polyautomata, however, are being extensively studied because of the relevance of certain subclasses of them to modeling living things. They have been particularly well investigated in the simplest forms—one dimension and no neighborhood other than the cell itself [26, 36, 58, 59, 74, 82, 84]—and are now being looked at in more generality [24, 25, 27, 35, 83, 85]. A detailed bibliography of this fastest growing branch of polyautomata theory is that of Lee and Rozenberg [55], containing over a hundred entries; an excellent text is that of Herman and Rozenberg [39]. Some of the biological implications of polyautomata theory have been explored at recent symposia attended by both biologists and polyautomata theorists [76, 77].

The cells in the von Neumann cellular space are *deterministic* because, for a given combination of present state and present input values, there is one and only one next state specified by the local transition function. A *nondeterministic* cell, however, has a local transition function that determines the next state only to within a set of possible choices. If this set is always of size one for a nondeterministic cell, than it is equivalent to a deterministic cell. Hence determinism is a special case of nondeterminism for cells. Hence, deterministic static polyautomata, those restricted to deterministic cells, are a subclass of the nondeterministic static polyautomata, which have nondeterministic cells. Whether the subclass is proper or not is an unanswered question, being related to the "lba problem", an unsolved problem of long standing in automata theory [9, 43, 93]. In the case of dynamic polyautomata, however, it has been shown that nondeterminism does lend additional power to that which can be obtained deterministically [35]. This is a consequence of the extension of the meaning of "(non)determinism" of dynamic polyautomata to sets of cells, as opposed to single cells in the static case.

The von Neumann cells are also examples of *Moore-type* finite-state machines that require a unit time step, the time between successive ticks of the clock, between any output and the inputs from which it is computed by the cell. A *Mealy-*

type cell permits zero delay between an output and its associated inputs. Thus a signal can pass instantaneously through a Mealy-type cell. In a Mealy-type polyautomaton, with cells of the Mealy-type, this implies that a signal can travel instantaneously to any cell or cells regardless of the distance as opposed to the situation in a Moore-type polyautomaton, with cells of the Moore-type, where a signal can travel instantaneously only from a cell to its neighbors. Holland [41] has proved that the Moore-type polyautomata are not as powerful as the Mealy-type polyautomata, there being finite "compositions", or interconnections, of (not necessarily identical) finite-state machines that can be simulated by the latter but not the former—i.e., Mealy-type polyautomata (in fact, uniform Mealy-type polyautomata) can be composition universal but Moore-type polyautomata cannot. Since a Moore-type cell is the special case of a Mealy-type cell where the property of instantaneous signal propagation is not used, Moore-type polyautomata are a sublclass of Mealy-type polyautomata. They form a proper subclass, by Holland's result.

The cellular spaces described in the discussion of self-reproduction are autonomous after time zero, having no input for external control. These are examples of *autonomous* polyautomata. If a polyautomaton is not autonomous, then an external input is assumed. Since external input schemes are numerous, the scheme of a particular subclass of polyautomata must be specified. For example, a single external input might be global, distributed to all cells, or local, distributed to a subset of cells, perhaps to only one cell. Or each cell might have an external input independent of the external inputs to each of the other cells. A finite polyautomaton may contain cells with inputs not connected to outputs of other cells in the polyautomaton nor designated as external inputs. Each of these is assumed to receive a constant *boundary signal*.

Similarly, a polyautomaton can have external output. Otherwise it is termed *without output* as, for example, the von Neumann cellular space. Again the possible output schemes are numerous and must be specified.

The polyautomaton described in this book is infinite, uniform, deterministic, Moore-type, autonomous, without output, and static[6]. Such a polyautomaton is called a *cellular space* if its cells lie on the nodes of the integer grid. It may have arbitrary dimensionality *d*. The von Neumann 2-dimensional cellular space has a 5-cell neighborhood, but, in general, the only restriction on neighborhood size in a cellular space (or any polyautomaton) is that it be finite. It has been shown, however, that the von Neumann neighborhood always suffices in the sense that a cellular space with arbitrary neighborhood can be simulated, time step for time step, by a cellular space with the von Neumann neighborhood after an initial encoding [91, 109]. This is true in any *d*-dimensional cellular space, where the *d*-dimensional von Neumann neighborhood consists of a cell and one nearest neighbor in each of the 2*d* directions from it.

It is an unproved claim, but apparently true, that the cost of neighborhood reduction is an increase in state-set size. If N is the size of a neighborhood and n is the size of the state set before neighborhood reduction, then the size of the

[6] [This is the usual meaning of the term CA.]

state set after reduction is on the order of n^N when reduction is to the Moore neighborhood (in d dimensions, a d-dimensional cube of 3^d cells centered on the cell that is in its own neighborhood). Of particular interest, because it is surprisingly difficult, is the cost of a reduction of the Moore neighborhood to the von Neumann. The best result known so far that holds for an arbitrary number of dimensions [91] shows an increase in the state-set size from n to n^V, where V is the volume (number of cells) in a d-dimensional sphere of radius $2d^{3/2}$. This cost has been reduced dramatically to n^4 for the 2-dimensional case [20] and to about n^{350} for the 3-dimensional [32]. Of course, for a particular cellular space or for a particular subclass of cellular spaces—e.g., the computation-universal cellular spaces—this cost can be greatly reduced, as already indicated.

Another surprising discovery has been that neighborhoods smaller than the von Neumann suffice. The neighborhood consisting of a cell and one nearest neighbor in each dimension works in a step-by-step simulation for arbitrary dimension, after an initial encoding [91]. It might appear that two-way information flow along each dimension is lost in the reduction of the von Neumann neighborhood to this neighborhood, but the trick is to let the configuration being simulated "slide" through the cellular space in time. In fact, the cell itself can be omitted from this neighborhood to obtain a d-cell neighborhood that also works, after an initial encoding. In a result even more counterintuitive than those above, Amoroso and Guilfoyle [5] have shown that, in two dimensions, any neighborhood can be reduced to a 3-cell neighborhood without using the trick of sliding, after an initial encoding. The cost is, in general, an increase to n^{45} states for reduction from the Moore neighborhood. This result has not yet been extended to higher dimensions. The phrase "after an initial encoding" can be omitted from all neighborhood reductions mentioned if a strict step-by-step simulation of the nonreduced cellular space is not necessary. This is because the cellular space itself can perform the initial encoding if appropriately designed.

Besides trading a state-set increase for a neighborhood reduction, it is also always possible to trade a neighborhood size increase for a reduction in the number of states. For N and n as before, the cost for reduction of n to m is an increase to on the order of $N \log_m n$ cells in the neighborhood, where m can be as small as 2 [91]. Again, this is a general result for an arbitrary cellular space and can be greatly improved in specific cases. It also has not been proved optimal.

Another valuable theoretical tool, in addition to the trade-off results above, is *speed-up*. One can design a cellular space to simulate any given cellular space, but k times faster, for any given integer k, after an initial encoding. Speed-up can be obtained by simply increasing the neighborhood, the number of cells "seen" instantaneously at each time step. More interesting, however, are speed-ups in which the neighborhood remains fixed. There is a cost, of course, and this is in state-set size—on the order of n^{k^d} for n as before and assuming the Moore neighborhood before and after speed-up [91]. Speed-up and neighborhood reduction can be accomplished simultanesouly, the cost being on the order of n^{kN}.

All the state-neighborhood trade-off results listed above for the cellular space also hold for a generalization to a polyautomaton called a *tessellation space* [107, 109], which is a cellular space with a single external input distributed to each cell.

Thus a cellular space is an autonomous tessellation space, or a tessellation space is a programmable cellular space, where the "program" is the sequence of externally supplied inputs. Each cell in a tessellation space can be thought of as having a finite set of local transition functions, and hence the tessellation space has a finite set of global transition functions. Each "instruction" in the program selects the global transition function to be used at that time step. The speed-up results for cellular spaces do not generalize to tessellation spaces unless it is assumed that the external program is one in which the instruction changes only once every k time steps, for speed-up by a factor k. Reference [109] is an excellent discussion of behavioural equivalence as well as structural equivalence of tessellation spaces, and hence cellular spaces.

A problem that is peculiarly tessellation, and not cellular, space theoretic is a nonautonomous generalization of the Garden-of-Eden question. This is called the *completeness problem* [108]: Is it possible to generate any finite configuration, with a suitable external program, where the initial configuration is *primitive*—one cell on, all others quiescent? If the answer is yes for a given subclass of tessellation spaces, then the subclass is said to be *complete*. The completeness problem has not been fully solved. For one-dimensional tessellation spaces with *contiguous* neighborhoods—i.e., such that every cell just to the left of a cell in a neighborhood is also in the neighborhood, except for the cell just left of the leftmost cell—the partial solutions are: Such tessellation spaces with 2-cell neighborhoods are not complete but with n-cell neighborhoods, $n \geq 3$, they are complete [60, 108]. There are only partial solutions for higher dimensions and neighborhoods that are not contiguous [61, 108].

Another variety of infinite polyautomata, closely related to the cellular and tessellation spaces, which has received detailed treatment in the literature is what I shall call an *iterative space*. An iterative space is a cellular space with one of its cells given an external input and an external output. Let this distinguished cell for each iterative space be called the *input-output cell*. Cole [23] has extensively analyzed the speed-up and neighborhood reduction possibilities of this subclass of polyautomata. His work was the inspiration for the analogous work for cellular spaces in [91]. The costs for speed-up and neighborhood reduction are higher for iterative spaces, though this has not been proved necessarily true. This is because of the special difficulties caused by the input-output cell. The best results known for neighborhood reduction from Moore to von Neumann neighborhoods in iterative spaces of two and three dimensions have state-set size costs of n^{21} and about n^{700}, respectively [32].

Because of its single input and single output, an interative space can be thought of as a "finite-state machine" with a highly structured, potentially infinite memory. Then, in analogy with finite-state machine theory, it becomes interesting to characterize the sets, or *languages*, of strings of input symbols that are accepted by the machine. An input string is *accepted*, if, just when or after the string has been completely entered, one symbol per time step, the output of the input-output cell goes into a specially designated *accept state*. Cole has characterized these languages for iterative spaces as a generalization of the languages accepted by finite-state machines. Kosaraju [50] has shown that the well-known

context-free languages [43] are a subset of the iterative space languages and are accepted by one-dimensional iterative spaces in a number of time steps proportional to the square of the length of the input string. He also showed that, by increasing the number of dimensions to two, the number of time steps for accepting the context-free languages could be reduced to a number proportional to the length of the input string. Seiferas [86] has proved that nondeterministic two-dimensional iterative spaces can accept in linear time any language accepted in linear time by a nondeterministic multihead Turing machine with a tape of arbitrary dimension. He also has shown that nondeterministic *d*-dimensional iterative spaces accept in linear time any language accepted in time n^d by this same type of Turing machine but with only one-dimensional tapes (*n* is the length of the input word).

A tessellation space has an external input like an iterative space and could be used similarly as a language acceptor if acceptance were defined in some way—e.g., by any nonquiescent cell going into an accept state. Similarly, a one-dimensional cellular space could be used as a language acceptor by assuming the initial pattern was the string to be processed for acceptance, which is defined perhaps as just suggested for tessellation spaces. Neither of these possibilities has been pursued, the sequence of configurations usually being of more interest in cellular and tessellation spaces than any one configuration. However, in the finite versions of these polyautomata, to be introduced next, language acceptance and recognition become very important, where a language is *recognized* if it is not only accepted but any string not in the language is *rejected* by a cell entering a special *reject state*. In dimensions higher than one, language recognition is sometimes referred to as *pattern recognition*.

A *cellular automaton* is a finite cellular space in which the finite set of cells is *neighbor-connected*. That is, the nodes representing any two cells in the interconnection graph of a cellular automaton are connected by a sequence of arcs. Similarly, a *tessellation automaton* is a finite neighbor-connected subset of cells in a tessellation space, and an *iterative automaton* is a finite neighbor-connected subset of cells in an iterative space, one of which must be the input-output cell. As already mentioned, for all these polyautomata, a cell with a missing neighbor has a special boundary signal substituted instead.

The pattern of states at time zero in a cellular automaton is taken to be the string or pattern to be accepted or recognized, where acceptance or rejection is determined by a specific cell, the *accept cell*, going into an accept or reject state. In one dimension this cell is, say, the rightmost cell. For the special subclass of two-dimensional cellular automata, each of which forms a rectangular array, the *rectangular* cellular automata, the accept cell is, typically, the northeast corner cell. For general two-dimensional cellular automata, it is, say the easternmost cell in the most northern row of cells. Language recognition by one-dimensional cellular automata is studied in [47] and [93]. Pattern recognition capabilities of rectangular cellular automata have been investigated by Beyer [18] and Kosaraju [51]. Pattern sets recognized by general two-dimensional cellular automata and the formal language characteristics of these sets have been described by Smith III [94]. These polyautomata are powerful pattern recognizers and fast, as might be

expected since the inherent parallelism is exploited. Interesting unsolved problems are: Is there a language that requires, of one-dimensional cellular automata, nonlinear recognition time—i.e., a number of time steps not proportional to the length of the initial pattern but larger? (The answer is yes for iterative spaces [23].) Can the context-free languages be accepted, by one-dimensional cellular automata, in *real time*—i.e., a number of time steps equal to the length of the initial pattern? What are the capabilities of cellular automata of dimension greater than two?

Nondeterministic cellular automata, which are the nondeterministic polyautomata obtained by replacing the deterministic cells of a cellular automaton with nondeterministic cells, have been useful in theoretical pattern recognition studies [93, 94]. For example, it has been proved that the context-free languages can be accepted in real time by nondeterministic one-dimensional cellular automata.

Tessellation automata have not been studied but, in a sense, iterative automata have. For real-time language recognition by a one-dimensional iterative space with the 3-cell nearest-neighbor neighborhood (which, because of the neighborhood reduction results, we can assume without loss of generality), the number of cells that can ever become nonquiescent, or used, in the iterative space is twice the length of the input string. Hence, so long as real-time recognition is the concern, the iterative space can be assumed finite in size, limited by the length of the input string, an iterative automaton. There are languages that can be accepted in real time by cellular automata which cannot be accepted in real time by iterative automata [93]. Hence cellular automata are inherently faster than iterative automata. This is not surprising since input is spatial to the former and temporal (hence one-way) to the latter.

A famous problem for cellular automata is the so-called Firing Squad Problem. Here each cell is called a “soldier” and one of them is called the “general”. All cells but one, the general, are off initially. Then the general gives the “command to fire”. The problem is to design the cells so that all go into the same, “firing”, state simultaneously and for the first time. The cell design must be independent of the number of soldiers. The problem has been solved numerous times in a sequence of improvements summarized in [65]. The solution, called the Firing Squad Theorem, has turned out to be quite useful in pattern recognition studies of cellular automata [93, 94]. A closely related result, also useful in pattern recognition, the Queen Bee Theorem solves a sort of reverse of the Firing Squad Problem [95]: Design a cellular automaton such that, if all cells are initially in the same state, one and only one of them is eventually, but rapidly, marked as the accept cell, the “queen bee”.

Although the Firing Squad Theorem was originally stated only for one-dimensional cellular automata, it has been generalized to rectangular cellular automata [18], to general cellular automata of arbitrary dimension [18, 71, 80, 88] and to certain node-static and dynamic polyautomata [38, 100]. In fact, Rosenstiehl [80] has shown that it remains valid even when a nonuniform interconnection scheme is allowed—that is, for finite, nonuniform, deterministic, Moore-type, autonomous, and static polyautomata without output—so long as the finite

set of cells is neighbor-connected and is input-output symmetric. This very interesting class of *cellular graph automata* has been analyzed in terms of what graph-theoretic properties of the interconnection graph each member of the class can decide or identify about itself [17, 81]. They have been called, in fact, "intelligent graphs". The decision or identification is said to have occurred when a *steady-state* configuration is obtained—that is, one that does not change under further applications of the global transition function. A far-reaching result of the work on cellular graph automata is that an independent global clock is unnecessary, that by appropriate cell design incorporating a local clock such a polyautomaton can synchronize itself. This applies, for instance, to cellular automata which are a special subclass of cellular graph automata. Some work has been done on cellular graph automata which are not input-output symmetric and which are not necessarily neighbor-connected. In the latter case, for example, it has been demonstrated that, in general, the polyautomaton cannot decide whether its interconnection graph is connected or not, except by judicious selection of initial configuration [81].

To my knowledge, the *tessellation graph automata*, each a cellular graph automaton with a global external input to all cells, have not been investigated. However, the *iterative graph automata*, each a cellular graph automaton with an input-output cell, have been studied in a manner quite different from the techniques so far discussed for polyautomata, namely, as "uniform modular realization, or decompositions" of finite-state machines (where "uniform" here means identical modules). Given a finite-state machine with, say, q states, the problem is to realize it with, or decompose it into, an interconnection of simpler, e.g., 2-state, finite-state machines which are identical. The input to the finite-state machine to be realized is distributed globally to all modules, or cells, in the *modular fsm realization*. Its output is taken from only one cell, the *output module*, and may be fed back to all cells as part of the global input. Any finite-state machine has a modular fsm realization with appropriate choice of module [10, 11, 99, 110]. Newborn and Arnold [70] have shown that any modular fsm realization can be replaced with an interative graph automaton—i.e., the external input can be restricted to the output module. An open question in this branch of polyautomata theory is: Is there a *linear* modular fsm realization of an arbitrary q-state finite-state machine—i.e., one that requires no more than cq cells, c constant? The restriction is that one module has to suffice for modular fsm realizations of the entire class of finite-state machines with p or fewer input and output values. Work has been performed on uniform as well as nonuniform modular fsm realizations [10, 11, 70] (where "uniform" again assumes its polyautomata theoretic meaning). An iterative graph automaton equivalent to a uniform modular fsm realization is, in general, not an interative automaton, so far as is currently known.

All of the polyautomata discussed so far have been static. I have already suggested that, with respect to the possible application of polyautomata theory, perhaps its most relevant subclass is the dynamic polyautomata that allow a cell to divide into one or more *daughter* cells and that allow the death, or disappearance, of cells. Consider a one-dimensional cellular automaton in which, at each time step, a cell not only changes state according to the states of its neighbors but

may also be replaced by a finite number of cells in specified states. Thus the "next state" is actually a string of states, but the next-state table is deterministic, as for the static case, if there is only one possible choice for the string in each distinct set of neighborhood circumstances. A cell in the finite cellular automaton so obtained is assumed to have the same neighborhood as before the cell division. Thus, if any cell's neighborhood is itself and its two nearest neighbors before division, then any cell's neighborhood after division is itself and its two nearest neighbors at that time. These one-dimensional *dynamic cellular automata* have been called "Lindenmayer systems" or "L-systems" after their originator Lindenmayer [57] who utilized them to model the growth of filamentary organisms. As mathematical entities, they have been extensively studied as "developmental systems" [36, 58, 59]. Just as for cellular and tessellation spaces, it is the sequence of configurations which is of interest, not any one configuration or the state of any one cell. If each pattern generated at each step is taken as a "word" in the "language" generated by these polyautomata, then the *developmental languages* generated by one-dimensional dynamic cellular automata with one-cell neighborhoods have been thoroughly described [58, 59, 82, 84]. Higher dimensional dynamic cellular automata have begun to receive treatment in the literature as well as the very interesting *dynamic cellular graph automata* [24, 25, 27, 50, 73]. This class should be the polyautomata most useful for modeling growth and development of living things. *Dynamic tessellation automata* have been investigated under the name of "table L-systems" [39, 82].

All of the polyautomata discussed so far have been Moore-type. By allowing Mealy-type cells, each variety of polyautomaton mentioned gives rise to a Mealy-type version. In particular, Holland [41] and Wagner [101] have studied Mealy-type cellular spaces. Finite Mealy-type polyautomata theory receives occasional investigation because of its close relationship to "real-world" computers [52, 96].

The bibliography which follows contains all references used in this survey. It is not meant to be exhaustive and should be combined with bibliographies such as that in Burks [19]. It does, however, contain many of the most recent papers written on polyautomata. Readings in this literature will familiarize interested persons in those areas I feel I have slighted or even ignored in this survey such as Mealy-type polyautomata [33, 34, 53], unbounded pattern growth in cellular spaces [22, 30], linear cellular automata [44, 45, 105], the regularity of one-dimensional pattern sets for cellular automata [49, 56], evolution of machines in cellular spaces [15, 16, 67], fault-tolerant cellular spaces [76] and pattern generation [42, 56, 98]. Applications of polyautomata theory include biological development [66, 75], highway traffic flow [75], physics [75], and economics [2].

Von Neumann states in his 1949 lectures (Part I of the following book), "You will see that our discussion of complex automata is very far from perfect and that one of our main conclusions is that we need very badly a theory which we do not at this moment possess." I suggest that polyautomata theory is just such a theory and hope that the many unsolved problems and open questions mentioned throughout this survey will inspire others to join in the exploration of the polyautomata theoretical space.

BIBLIOGRAPHY

Abbreviations used frequently in the following are:
- ACM - Association for Computing Machinery
- I and C - Information and Control
- IEEE - Institute of Electrical and Electronic Engineers
- IEEETC - IEEE Transactions on Computers
- JACM - Journal of the ACM
- JCSS - Journal of Computer and Systems Sciences
- MIT - Massachusetts Institute of Technology

[1] Aladyev, V. Z. On the Theory of the Homogeneous Structures, Technical Report, Institute of Experimental Biology, Academy of Sciences of the Estonian SSR, USSR, 1972 (in Russian).

[2] Albin, Peter S. Cellular automata representation of economic models. N.Y.U. Graduate School of Business Administration Working Paper #72-05, 1972.

[3] Amoroso, Serafino, and Cooper, Gerald. The garden-of-eden theorem for finite configurations. Proceedings of the American Mathematical Society **44**: 189-197, 1970.

[4] Amoroso, Serafino, and Cooper, Gerald. Tessellation structures for reproduction of arbitrary patterns. JCSS **5**: 455-464, 1971.

[5] Amoroso, Serafino, and Guilfoyle, R. Some comments on neighbourhood size for tessellation automata. I and C **21**, 48-55, 1972.

[6] Amoroso, Serafino, and Patt, Yale Nance. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. JCSS **6**, 448-464, 1972.

[7] Arbib, Michael A. Simple self-reproducing universal automata. I and C **9**, 177-189, 1966.

[8] Arbib, Michael A. Self-reproducing automata—some implications for theoretical biology, pp. 204-226 of Towards a Theoretical Biology, Volume 2: Sketches, editor C. H. Waddington, Chicago: Aldine, 1969.

[9] Arbib, Michael A. Theories of Abstract Automata. Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

[10] Arnold, Thomas F., and Newborn, Monroe M. Iteratively realized sequential circuits: further considerations. Conference Record of the 10th Annual IEEE Symposium on Switching and Automata Theory **10**: 194-212, 1969.

[11] Arnold, Thomas F., Tan, C. J., and Newborn, Monroe N. Iteratively realized sequential circuits. IEEETC **C-19**: 54-66, 1970.

[12] Atrubin, A. J. A one-dimensional real-time iterative multiplier. IEEETC **EC-14**: 394-399, 1965.

[13] Balzer, Robert M. An 8-state minimal time solution to the firing squad synchronization problem. I and C **10**: 22-42, 1967.

[14] Banks, E. Roger. Cellular Automata. AI Memo No. 198, MIT Artificial Intelligence Lab, 545 Technology Square—Room 821, Cambridge, Massachusetts 02139, 1970.

[15] Barricelli, N. A. Symbiogenetic evolution processes realized by artificial methods. Methodos **9**, 148-182, 1957.

[16] Barricelli, N. A. Numerical testing of evolution theories. Acta Biotheoretica **16:** 69-126, 1963.

[17] Berstel, Jean. Quelques application des reseaux d'automates a des problemes de la theorie des graphes. Doctorat 3d Cycle thesis, Institut de Programmation, Universite de Paris, June, 1967.

[18] Beyer, Terry. Recognition of Topological Invariants by Modular Arrays. AI Memo No. 166, MIT Artificial Intelligence Lab, 545 Technology Square—Room 821, Cambridge, Massachusetts 02139, 1968.

[19] Burks, Arthur W., editor. Essays on Cellular Automata. University of Illinois Press, Urbana, Illinois, 1970.

[20] Butler, Jon T. A note on cellular automata simulations. I and C, to appear, 1974.

[21] Case, James H., and Stewart, Neil C. Short Wire Theory I. Catalogued into the Repository of the ACM, January 3, 1966. The authors are with the Department of Math., University of Utah, Salt Lake City, Utah.

[22] Codd, Edgar Frank. Cellular Automata. ACM Monograph Series. Academic Press, Inc. New York, 1968.

[23] Cole, Stephen N. Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEETC **C-18**: 349-365, 1969.

[24] Culik II, Karel. Weighted growth-functions of D0L-systems and growth functions of parallel graph rewriting systems. Research Report CS-74-24, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1974. (Abstract in [77]).

[25] Culik II, Karel, and Lindenmayer, Aristid. Parallel rewriting on graphs and multidimensional development, Research Report CS-74-22, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1974.

[26] Dalen, D. van. A note on some systems of Lindenmayer. Mathematical Systems Theory **5**: 128-140, 1971.

[27] Ehrig, Hartmut, and Kreowski, H. J. Parallel graph grammars. Abstract in [77].

[28] Fischer, Patrick C. Generation of primes by a one-dimensional real-time iterative array. JACM **12**: 388-394, 1965.

[29] Gardner, Martin. The fantastic combinations of John Conway's new solitaire game "life". Mathematical Games Department. Scientific American **223**: 120-123, 1970.

[30] Gardner, Martin. On cellular automata, self-reproduction, the Garden of Eden and the game "life". Mathematical Games Department. Scientific American **224**: 112-117, 1971.

[31] Golze, Ulrich. Some new differences between 1- and 2- dimensional cellular spaces. Article in this volume.

[32] Hamacher, V. C. Machine complexity versus interconnection complexity in iterative arrays. IEEETC **C-20**: 321-323, 1971.

[33] Hennie III, Frederick C. Analysis of bilateral iterative networks. Transactions of the Institute of Radio Engineers Professional Group on Circuit Theory **CT-6**: 35-45, 1959.

[34] Hennie III, Frederick C. Iterative Arrays of Logical Circuits. The MIT Press and John Wiley Sons, Inc., New York, 1961.

[35] Herman, Gabor T., Closure properties of some families of languages associated with biological systems. I and C **24**: 101-121, 1974.

[36] Herman, Gabor T. The computing ability of a developmental model for filamentous organisms. Journal of Theoretical Biology **25**: 423-435, 1969.

[37] Herman, Gabor T. On universal computer-constructors. Information Processing Letters **2**: 61-64, 1973.

[38] Herman, Gabor T., Liu, Wu-Hung, Rowland, Stuart, and Walker, Adrian. Syncchronization of growing cellular arrays. I and C, to appear 1974.

[39] Herman, Gabor T. and Rozenberg, Grzegorz. Developmental Systems and Languages. North-Holland Publishing Company, Amsterdam, 1975.

[40] Holland, John H. Iterative circuit computers. Pp. 259-265 of Proceedings of the 1960 Western Joint Computer Conference, 1960. (Essay 13 of Burks [19]).

[41] Holland, John H. Universal embedding spaces for automata, pp. 223-243 of Cybernetics of the Nervous System (Progress in Brain Research 17), editors Norbert Wiener and J. P. Schade. New York: Elsevier, 1965 (cf. Essay 15 of Burks [19]).

[42] Holladay, J. C., and Ulam, Stanislaw M. On some combinatorial problems in patterns of growth, I. Notices of the American Mathematical Society **7**: 234, 1970.

[43] Hopcroft, John, and Ullman, Jeffrey D. Formal Languages and their Relation to Automata. Addison-Wesley, Reading, Massachusetts, 1969.

[44] Hu, Ming-Kuei, and Iosupovicz, Alexander. Analysis of the terminal behaviour of some classes of iterative arrays of linear machines. IEEETC **C-21**: 1394-1397, 1972.

[45] Iosupovicz, Alexander, and Hu, Ming-Kuei. A class of autonomous one-dimensional iterative arrays of linear machines. IEEETC **C-21**: 1073-1086, 1972.

[46] Jump, J. Robert, and Kirtane, Jayant S. On the interconnection structure of cellular networks. I and C **24**: 74-91, 1974.

[47] Kasami, T., and Fujii, M. Some results on capabilities of one-dimensional iterative logical networks. Electronics and Communications in Japan **51-C**: 167-176, 1968.

[48] Kilmer, William L. On dynamic switching in one-dimensional iterative logic networks. I and C **6**: 399-415, 1963.

[49] Kobuchi, Y., and Nishio, H. Properties of certain state sets in the system of one-dimensional iterative automata (in Japanese). Electronics and Communications in Japan **54-C**: 396-403, 1971. (Revised in Journal of Information Science **5**: 199-216, 1973).

[50] Kosaraju, S. Rao. Speed of recognition of context-free languages on cellular arrays. Society of Industrial and Applied Mathemaatics (SIAM) Journal on Computing, to appear, 1974.

[51] Kosaraju, S. Rao. On some open problems in the theory of cellular automata. IEEETC, to appear, 1974.

[52] Kruse, Bjorn. A parallel picture processing machine. IEEETC **C-22**: 1075-1087, 1973.

[53] Kukrija, S. N., and Chen, I.-N. Combinational and sequential cellular structures. IEEETC **C-22**: 813-823, 1973.

[54] Lee, Chester Y. Synthesis of a cellular computer, pp. 217-234 of Applied Automata Theory, editor J. T. Tau. New York: Academic Press, 1968.

[55] Lee, K. P., and Rozenberg, Grzegorz. Bibliography on developmental systems (in the sense of L-systems). Report No. 3, Universiteit Antwerpen, U.I.A., Department of Mathematics, Fort VI Straat, 2610 Wilrijk, Belgium, 1974.

[56] Lieblein, E. A Theory of Patterns in Two-Dimensional Tessellation Space. Ph.D. dissertation, University of Pennsylvania, Philadelphia, Pennsylvania, 1968.

[57] Lindenmayer, A. Mathematical models for cellular interactions in development. Parts I and II. Journal of Theoretical Biology **18**: 280-315, 1968.

[58] Lindenmayer, A. Developmental systems without cellular interactions, their languages and grammars. Journal of Theoretical Biology **30**: 455-484, 1971.

[59] Lindenmayer, A. and Rozenberg, Grzegorz. Developmental systems and languages. Conference Record of the 4th Annual ACM Symposium on Theory of Computing **4**: 214-221, 1972.

[60] Maruoka, Akira and Kimura, Masayuki. Completeness problem of one-dimensional binary scope-3 tessellation automata. JCSS, to appear, 1974.

[61] Maruoka, Akira, and Kimura, Masayuki. Completeness problem of multi-dimensional tessellation automata. Submitted for publication, 1975.

[62] Maugh III, Thomas H. Molecular biology: a better artificial gene. Research News. Science **181**: 1235, 1973.

[63] Merzinich, Wolfgang. Cellular automata with additive local transition, pp. 186-194 of Proceedings of the First International Symposium: Category Theory Applied to Computation and Control, Department of Computer and Information Sciences, University of Massachusetts, Amherst, Massachusetts 01002, 1974.

[64] Moore, Edward F. Machine models of self-reproduction. Proceedings of Symposia in Applied Mathematics **14**: 17-33, 1962. (Essay 6 of Burks [19]).

[65] Moore, F. R., and Langdon, G. C. A generalized firing squad problem. I and C **12**: 212-220, 1968.

[66] Mortimer, J. A. A cellular model for mammalian cerebellar cortex. Technical Report No. 03296-7-T, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan, 1970.

[67] Myhill, John. The abstract theory of self-reproduction, pp. 106-118 of Views on General Systems Theory, Proceedings of the Second Systems Symposium at Case Institute of Technology, editor Mihajlo D. Mesarovic, 1964. (Essay 8 of Burks [19]).

[68] Myhill, John. The converse of Moore's Garden-of-Eden theorem. Proceedings of the American Mathematical Society **14**: 685-686, 1963 (Essay 7 of Burks [19]).

[69] Nagl, Manfred. On a generalization of Lindenmayer-systems to labelled graphs. Article in this volume.

[70] Newborn, Monroe M., and Arnold, Thomas F. Universal modules for bounded signal fan-out synchronous sequential circuits. IEEETC **C-21**: 63-79, 1972.

[71] Nguyen, H. B., and Hamacher, V. C. Pattern synchronization in two-dimensional cellular spaces. I and C, to appear, 1974.

[72] Ostrand, Thomas J. Pattern reproduction in tessellation automata of arbitrary dimension. JCSS **5**: 623-628, 1971.

[73] Paz, Azaria, Multidimensional parallel rewriting systems. Article in this volume.

[74] Paz, Azaria, and Salomaa, Arto. Integral sequential word functions and growth equivalence of Lindenmayer systems. I and C **23**: 313-343, 1973.

[75] Pilgrim, P. C., Bazaj, S., Golze, U., and McConnell, J. Cellular space models for particle physics, biological development, and highway traffic flow: some initial explorations. Edited by B. P. Zeigler. Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan, 1972.

[76] Proceedings of the Conference of Biologically Motivated Automata Theory. The MITRE Corporation. McLean, Virginia, 1974.

[77] Proceedings of the Conference on Formal Languages, Automata, and Development. Noordwijkerhout, The Netherlands, 1975.

[78] Richardson, Daniel. Tessellations with local transformations. JCSS **6**: 373-388, 1972.

[79] Rosenfeld, Azriel, and Milgram, David L. Parallel/sequential array automata. Information Processing Letters **2**: 43-46, 1973.

[80] Rosenstiehl, Pierre. Existence d-automate finis capables de s'accorder bien qu'arbitrairement connectes et nombreaux. Bulletin of the International Computation Centre **5**: 245-261, 1966.

[81] Rosenstiehl, Pierre, Fiksel, J. R., and Holliger, A. Intelligent graphs: networks of finite automata capable of solving graph problems, pp. 219-265 of Graph Theory and Computing, editor Ronald C. Read. New York: Academic Press, 1972.

[82] Rozenberg, Grzegorz. T0L systems and languages. I and C **23**: 357-381, 1973.

[83] Rozenberg, Grzegorz. L-systems with interactions. JCSS, to appear, 1974.

[84] Rozenberg, Grzegorz, and Doucet, P. G. On 0L languages. I and C **19**: 302-318, 1971.

[85] Rozenberg, Grzegorz, and Lee, K. P. Developmental systems with finite axiom sets. Part II. Systems with interactions. International Journal of Computing Mathematics, to appear, 1974.

[86] Seiferas, Joel I. Observations on nondeterministic multidimensional iterative arrays. Conference Record of the 6th Annual ACM Symposium on Theory of Computing **6**: 276-289, 1974.

[87] Selkow, S. M. One-pass complexity of digital picture properties. JACM **19**: 283-295, 1972.

[88] Shinahr, Ilka. Two- and three-dimensional firing-squad synchronization problems. I and C **24**: 163-180, 1974.

[89] Smith III, Alvy Ray. Simple computation-universal cellular spaces and self-reproduction. Conference Record of the 9th Annual IEEE Symposium on Switching and Automata Theory **9**: 269-277, 1968.

[90] Smith III, Alvy Ray. Cellular Automata Theory. Technical Report No. 2, Digital Systems Lab, Stanford University, Stanford, California, 1969.

[91] Smith III, Alvy Ray. Cellular automata complexity trade-offs. I and C **18**: 466-482, 1971.

[92] Smith III, Alvy Ray. Simple computation-universal cellular spaces. JACM **18**: 339-353, 1971.

[93] Smith III, Alvy Ray. Real-time language recognition by one-dimensional cellular automata. JCSS **6**: 233-253, 1972.

[94] Smith III, Alvy Ray. Two-dimensional formal languages and pattern recognition by cellular automata. Conference Record of the 12th Annual IEEE Symposium on Switching and Automata Theory 12: 144-152, 1971 (Also, Pattern recognition by cellular automata, to be published, JCSS.[7])

[95] Smith III, Alvy Ray. The Queen Bee Theorem: a desynchronization theorem for cellular automata. Submitted for publication[8].

[96] Sturman, Joel N. An interatively structured general-purpose digital computer. IEEETC C-17: 2-9, 1968.

[97] Thatcher, James W. Universality in the von Neumann Cellular Model. Technical Report 03105-30-T, ORA, University of Michigan, Ann Arbor, Michigan 1964. (Essay 5 of Burks [19]).

[98] Ulam, Stanislaw M. On some mathematical problems connected with patterns of growth of figures. Proceedings of Symposia in Applied Mathematics 14: 215-224, 1962. (Essay 9 of Burks [19]).

[99] Ullman, Jeffrey D., and Weiner, Peter. Uniform synthesis of sequential circuits. Bell System Technical Journal **48**: 1115-1127, 1969.

[100] Varshavsky, V. I., Marahkovsky, V. B., and Pechansky, V. A. Synchronization of interacting automata. Mathematical Systems Theory **4**: 212-230, 1970.

[101] Wagner, Eric G. An Approach to Modular Computer, I: Spider Automata and Embedded Automata. IBM Research Report RC 1107, IBM T. J. Watson Research Center, Yorktown Heights, New York, 1964.

[102] Wagner, Eric G. Modular Computers II: Graph Theory and the Interconnection of Modules. IBM Research Report RC 1414, IBM T. J. Watson Research Center, Yorktown Heights, New York, 1965.

[103] Wainwright, Robert, editor. Lifeline (a newletter devoted to the game of "life") **3**: 31, 1971, 1280 Edorio Road, Yorktown Heights, New York 10598.

[104] Waksman, Abraham. An optimum solution to the firing squad synchronization problem. I and C **9**: 67-78, 1966.

---

[7] [I changed careers at this point, to computer graphics, and never proceeded to publish this paper in the JCSS.]

[8] [Similarly never completed.]

[105] Winograd, Terry. A Simple Algorithm for Self-replication. AI Memo No. 197, MIT Artificial Intelligence Lab, 545 Technology Square—Room 821, Cambridge, Massachusetts 02139, 1970.

[106] Yaku, Takeo. The constructibility of a configuration in a cellular automaton. JCSS **7**: 481-496, 1973.

[107] Yamada, Hisao, and Amoroso, Serafino. Tessellation automata. I and C **14**: 299-317, 1969.

[108] Yamada, Hisao, and Amoroso, Serafino. A completeness problem for pattern generation in tessellation automata. JCSS **4**: 137-176, 1970.

[109] Yamada, Hisao, and Amoroso, Serafino. Structural and behavioural equivalences of tessellation automata. I and C **18**: 1-31, 1971.

[110] Weiner, Peter, and Hopcroft, John E. Bounded fan-in, bounded fan-out uniform decompositions of synchronous sequential machines. Proceedings of the IEEE **56**: 1219-1220, 1968.