EVOLVING A REPLICATOR
The Genetic Programming of Self Reproduction in Cellular Automata

Hugo de Garis

(Current Address)
Brain Builder Group, Evolutionary Systems Department,
ATR Human Information Processing Research Laboratories,
2-2 Hikari-dai, Seika-cho, Soraku-gun,
Kansai Science City, Kyoto, 619-02, Japan.
tel: + 81 7749 5 1079, fax: + 81 7749 5 1408,
email: degaris@hip.atr.co.jp

(Work performed mainly at)
Computational Models Section, Computer Science Division,
Electro Technical Lab (ETL),
1-1-4 Umezono, Tsukuba Science City, Ibaraki, 305, Japan.
tel : + 81 298 58 5870, fax : + 81 298 58 5871
email : degaris@etl.go.jp

Keywords

Genetic Programming (GP), Genetic Algorithms (GAs), Replicators, Self-Reproduction, Nanotechnology, Scanning Tunneling Microscopes (STMs), Atomic Positioning, Molecular Replicators, Nanots (Nano Scale Robots), Cellular Automata, Partial Replication, Artificial Life, Quantum Dot Arrays, Quantum-Electronic Computers, Molecular Dynamics, Software-Based Evolution of Molecular Replicators, CA Neurite Networks, CA Neurons, Darwin Machines.

Abstract

This paper presents the results of an investigative study into the evolution of cellular automata replicators using Genetic Programming (GP) techniques (i.e. using Genetic Algorithms (GAs) to build/evolve complex systems). There are at least two reasons why such a study might be considered interesting. One reason is to explore how difficult the *evolution* of (CA) replicators might be, a topic of importance for Artificial Life. Another reason is the possibility that the evolution of CAs, if successful, may provide tools for next-generation quantum-electronic computers (e.g. using "quantum dot arrays") which may use CAs as their operating principle.

1. Introduction

One of the growing intellectual interests of the author is nanotechnology [Drexler 1992], which the author believes will prove in the long term to be the enabling technology for truly effective artificial life forms, and true artificial intelligence. However, since nanotech, by definition, is molecular scale engineering, if ever its products are to be useful to human beings, they will have to be of human size, so the number of nano scale robots used to build these products will have to be of the order of Avogadro's number, i.e. a trillion trillion. The state of the art in nanotech at the time of writing is using Scanning Tunneling Microscopes (STMs) to position individual atoms into desired patterns on a metal surface [Eigler & Schweizer 1990]. It is likely that within a year or two, chemical reactions between such positioned atoms will be possible. Once this breakthrough occurs, an explosion in the number of possible substances which can be built at an atomic scale will result. Hence it is not surprising that world wide interest in nanotech has risen very sharply over the last two years.

But, one cannot position an Avogadro number of atoms, one atom at a time, to build a human scale device, because it would take years to assemble one mole of material. Somehow, nano scale robots (nanots) will have to reproduce a large number of copies of themselves (the way biological cells do) before commencing the construction process. The author got interested in the question of replication, because if it becomes possible to build an artificial molecular replicator by positioning only a small number of atoms (i.e. less than several thousand), then STM positioning (plus chemical reactions) may prove to be a feasible route towards this goal. Once the first replicator is built, it can then be used to reproduce an Avogadro number of copies. These replicators would also need to be functional of course, otherwise there would be no point in building them.

Since the author works with computers and electronics, and not with STMs nor molecular biology, his thoughts turned to how it might be possible to build/evolve replicators in an abstract sense, hoping that ideas that might result from a successful evolution of abstract replicators, might be carried over to the molecular domain and STMs. It is also possible that the evolution of abstract replicators might be interesting and useful in its own right. For example, it might be possible that generic hardware devices might be able to configure themselves by using electronic replicating patterns. Next-generation quantum-electronic devices with dimensions so small that quantum tunneling becomes their essential operating principle, will necessitate that these devices operate as cellular automata, because only near neighbor interactions will be feasible. Next-generation computers based on such devices (e.g. quantum dot arrays [Kirk & Reed 1992]) may find techniques for the evolution of cellular automata very useful.

The author therefore set out to investigate how interesting, difficult or even feasible it might be, to evolve replicators. The framework in which the author chose to conduct the experiments presented below, was that of cellular automata (CAs) [Wolfram 1986]. Besides nanotech, one other stimulus for writing this paper came from reading a paper by Byl [Byl 1989] which presented a hand-crafted CA with only 12 cells and 6 states which was capable of replication. Byl's work was derived from similar work of Langton's [Langton 1984], who hand-crafted a CA replicator with roughly 70 cells and 8 states. The immediate thought of the author, was that that number seemed small enough to make it possible to evolve such a replicator, using GP techniques. The search space might not be gargantuan.

Firstly, some introductory definitions. A cellular automaton (CA) is an abstract entity (usually called a "cell") which is positioned on a grid (usually 1, 2 or 3 dimensions) which can be in any one of a finite number of states S. All the cellular automata on the grid update their states synchronously, by applying state update rules. The next state of a cell depends upon its current state and the current states of its neighboring cells. These neighboring cells of a cell (in a 2D grid) are usually the 4 cells to the E, N, W, and S, or the 8 cells to the E, NE, N, NW, W, SW, S, and SE. Hence the cell at coordinates (i, j) with a 4 cell neighborhood, will update its state according to the rule :-

$$S_{t+1}(i, j) = f[S_t(i, j), S_t(i+1, j), S_t(i, j+1), S_t(i-1, j), S_t(i, j-1)]$$

If there are N possible states and the neighborhood has 4 cells, then there are N to the power N^5 possible sets of state transition rules. Each set contains N^5 rules. The notation used for the state transition rules (i.e. CtRbL--->C) is a list of states of the Center cell, its top cell, its Right-hand cell, its bottom cell, its Left-hand cell, and the next state of the Center cell. With the smallness of Byl's replicator in mind, the author set himself the task of exploring the possibility or otherwise of using GP techniques to build/evolve a CA replicator.

2. Experimental Structures

When trying to evolve anything using GAs, there are several important considerations which must be dealt with. One is the representation of the chromosome used (to determine the behavior of the CA). Another is the fitness definition of the

performance specified by the chromosome. These two considerations will now be discussed in detail. A CA needs to be given a starting configuration, i.e. each cell in the grid needs to know its initial state. Hence the chromosome may need to specify some starting configuration. The CA also needs to know its state repertoire and its state transition rules. Hence the chromosome may need to specify the number of states and the state transition rules. Since it is extremely unlikely that an initial random set of transition rules will replicate an initial configuration, there needs to be some form of "partial replication" fitness definition. Finding this "partial replication" fitness definition was one of the more difficult aspects of the work presented in this paper.

With the above considerations in mind, the author began thinking about how to evolve a CA replicator. The approach taken was the reverse of Byl's, i.e. instead of simplifying Byl's ideas to produce an even simpler model (as Byl did to Langton's model), the author reversed this top-down approach, to a bottom-up approach, by starting at the rock-bottom, absolute simplest CA replicator, and working up to a complexity level which was thought to be small enough to be evolvable, but complex enough not to be considered trivial. As will shortly be seen, this compromise was not an easy one to make.

```
  0          00          000
010   -->  0110   -->  01110   -->  etc.
  0          00          000
```

State Transition Rules

10000-1	00100-0
00010-0	10100-1
00001-1	10101-1
01000-0	10001-1

FIG. 1 A TWO-STATE Cellular Automata REPLICATOR

FIG. 1 shows the simplest possible CA (arithmetic) replicator, which consists of cells with 2 possible states (0 and 1), a starting configuration of one cell in state 1 and all the other cells in state 0, and the few state transition rules as shown. Note that arithmetic replicators increase arithmetically in number (1,2,3,4), whereas geometric replicators increase geometrically in number (1,2,4,8). FIG. 2 shows a simple 3 state CA arithmetic replicator, whose replication occurs over two cycles. The state transition rules are not shown. FIG. 3 shows a simple four-state CA arithmetic replicator, whose replication occurs over three cycles. The state transition rules are not shown.

```
  00          000          0000          00000
0120   -->  02120   -->  012120   -->  0212120   -->  etc.
  00          000          0000          00000
```

FIG. 2 A THREE-STATE Cellular Automata REPLICATOR

```
                                          0
  0          00          000          0030
 030        0120         02310        03120
0120   -->  0230   -->  03100   -->  01200   -->  etc.
  00          00          00          00
```

FIG. 3 A FOUR-STATE Cellular Automata REPLICATOR "Staircase"

At this stage, the author began to be concerned with the size of the search space and hence the length of the chromosome necessary to evolve the state transition rules. In a two dimensional CA grid, with a four state, four cell neighborhood, there are 4^5 = 1024 possible left hand sides (LHSs) for these state transition rules. For each left hand side, there are four possible new states. The chromosome would thus need to

code for one of these four possible new states for all 1024 possible left hand sides. This is already not a small chromosome. If there are 5 possible states in the CA, with a four cell neighborhood, then there are $5^5 = 3125$ possible LHSs for these state transition rules. With 6 possible states, there are 7776 LHSs. Using a monoprocessor computer, such a chromosome length is considered to be rather large for Genetic Algorithms. Hence the author was limited to evolving CA replicators with 6 or less states. In practice, the author began with 4 states and 4 neighbors. The 8 neighbor case was not even considered, for obvious reasons.

The author was then confronted with several questions. Would the evolution need to be closely supervised, i.e. would a target configuration need to be supplied for each cycle, or would only a final configuration for the last cycle be needed? Could the evolution be less supervised? How does one define a fitness definition for a potential replicator? We begin with a suggested fitness definition. FIG. 4 shows how.

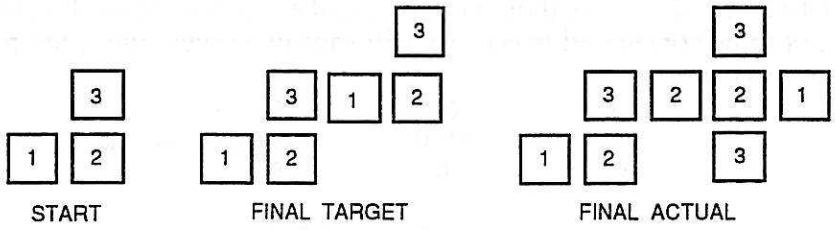


FIG. 4 DEFINING THE FITNESS

The fitness definition chosen is expressed in the formula :-

$$\text{Fitness} = 100*\text{NCOC} + 144 - \text{NNZC}$$

where :- NCOC is the Number of Correctly Ordered Cells
144 (i.e. 12*12) is the number of cells in the grid whose states can change
NNZC is the Number of Non Zero state Cells

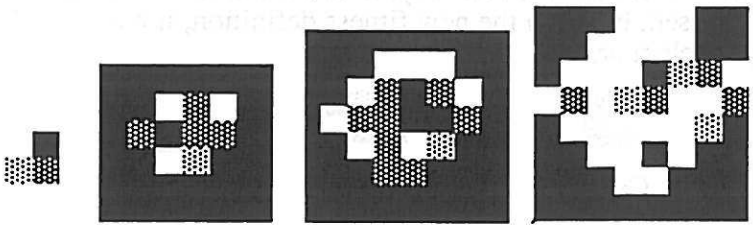
For example, the fitness of the final actual configuration in FIG. 4 would be 100*3 + 144 - 8 = 436. Note that there are only 3 "correctly ordered cells" (i.e. as counted from the bottom left of the final target configuration, and following its "path"). Note that the state "2" cell, with state "3" cells above and below it in the final actual configuration, has the same state as the corresponding cell in the final target configuration, but this final actual configuration cell does not contribute to NCOC because the cell to its left (also a state "2" cell) does not have the same state ("1") as its corresponding cell in the final target. The fitness of the final target configuration would be 6*100 + 144 - 6 = 738. This fitness definition, with its weighting of 100, was chosen to strongly favor the factor NCOC, and to mildly favor a decrease in NNZC. The 144 forced the fitness to remain positive, because early chromosomes produced many non zero cells. Choosing this fitness definition meant that whenever NCOC increased by 1, the fitness jumped significantly, and the corresponding chromosome soon dominated the evolving population. Once the desired sequence of target cells appeared in the final actual configuration, the fewer the non zero state cells, the higher the fitness, thus removing "stray" (i.e. undesired non zero state) cells. Using a population of 50 chromosomes, the 1024 slots on the chromosome were filled with digits (0,1,2, or 3), rather than two bits. It was felt that using two bits per slot would create problems. For example, if a slot specifies that the next state should be a "3", but ought to be a "0", then two (single bit) mutations would be needed to convert the "0" to a "3". The intermediate "1" or "2" (as specified by one non-zero bit) may decrease the fitness of the chromosome and be eliminated. Therefore it was felt that a direct mutation from "0" to "3", using integer representation rather than bits would be advisable. A state transition rule of the form $C_i\text{TRBL} \rightarrow C_{i+1}$ was represented on the chromosome

by placing the next state (i.e. the digit C_{i+1} which took an integer value of 0,1,2 or 3) in the Pth. slot position, where P was calculated from the formula :-

$$P = 256*C_i + 64*T + 16*R + 4*B + L$$

3. Experimental Results

In the first experiment, the initial (start) and target configurations were as shown in FIG. 3. Using a mutation rate of 0.003 per slot per chromosome, and uniform crossover, the following results, as shown in FIG. 5, were obtained from the elite chromosome, for 3 cycles, over 2500 generations. The fitness was measured after cycle 3. The states (0, 1, 2, 3) correspond to (white, light gray, dark gray, black) respectively.

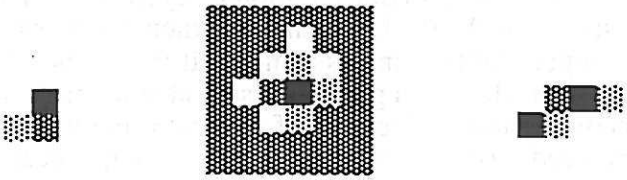


Cycle 0 Cycle 1 Cycle 2 Cycle 3

FIG. 5 STATES AT CYCLES 0, 1, 2, 3

Cycle 0 Cycle 1

FIG. 6 1 CYCLE EVOLUTION OF FIG. 3 REPLICATOR



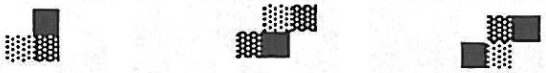
Cycle 0 Cycle 1 Cycle 2

FIG. 7 2 CYCLE (SHAPED) EVOLUTION OF FIG. 3 REPLICATOR

The third cycle configuration of the elite chromosome shown in FIG. 4 evolved after 1000 generations but did not change at all for a further 1500 generations. The evolution had either got stuck, or, as will become evident later on in this paper, there was some other problem. Note that the edge cells of the 12*12 cell matrix used in these experiments were permanently set to state "0". They could not change their state. The direct approach, as shown in the above figure, was considered disappointing and unsatisfactory. Therefore a more rigorous, more tightly supervised approach was adopted in order to force the evolution in desired directions. It was decided to evolve the replicator shown in FIG. 3 by using "shaping" techniques [e.g. de Garis 1990], i.e. splitting the evolution into phases, where the resulting chromosome population of one phase of evolution is used as the starting population in a second phase. For example, the chromosome population resulting from a successful 1 cycle evolution is used as the starting population for a 2 cycle evolution.

FIG. 6 shows the results of a successful 1 cycle evolution of the replicator of FIG. 3, using 100 generations. The resulting chromosome population was input into a second phase of evolution over two cycles, where the target configuration for 2 cycles is shown in FIG. 3. FIG. 7 shows the results of the elite chromosome after 500 generations, for 2 cycles, using the chromosome population resulting from FIG. 6. Note that the configuration at the first cycle is no longer what it was in FIG. 6. This was thought to be unsatisfactory, so a further change in approach was necessary.

To force the evolution to generate the desired target configurations from one cycle to the next, a change in the fitness definition was employed. For a 2 cycle evolution, the fitness value from the first cycle (for a single chromosome) was multiplied by 1000 and then added to the fitness value from the second cycle, e.g. if the 1 cycle fitness is 540 (i.e. 100*4 + 144 - 4) and the 2 cycle fitness is 639 (i.e. 100*5 + 144 - 5), then the fitness for the full 2 cycle evolution is 540639 (1000*540 + 639). Shaping was also used, but with the new fitness definition, it may only have served as an evolutionary accelerator.



Cycle 0 Cycle 1 Cycle 2

FIG. 8 2 CYCLE EVOLUTION WITH WEIGHTED FITNESS DEFINITION

Fig. 8 shows the results of the elite chromosome after 1200 generations, for 2 cycles, using the new fitness definition. This time the first cycle configuration was correct, but the second cycle configuration never managed to produce the target configuration, no matter how many generations were allowed for the evolution.

The desired fifth (state "1") cell to the right of the fourth (state "3") cell never appeared in the second cycle. The suspicion arose that perhaps such a target configuration might be logically impossible. A detailed analysis soon revealed that such a logical impossibility was in fact the case. To see why, consider the blank (state "0") cell to the right of the state "2" (dark gray) cell in the cycle 0 configuration. Its state transition rule (using the CTRBL-C convention) is 00002-0. Consider now the blank (state "0") cell to the right of the rightmost state "2" (dark gray) cell in the cycle 1 configuration. To attain the target configuration of cycle 2 (as shown in FIG. 3), its state transition rule should be 00002-1, which contradicts the above 00002-0.

This logical impossibility came as a shock. It meant that not all configuration sequences are possible, which in turn places considerable restrictions upon what can be evolved in a supervised manner. The task of replicator evolution is thus made a lot more difficult. To proceed further, another change in strategy became necessary. It was decided to change the replicator pattern from a "staircase" to an "arrow" as shown in FIG. 9. Perhaps a different pattern might avoid the logical impossibility.

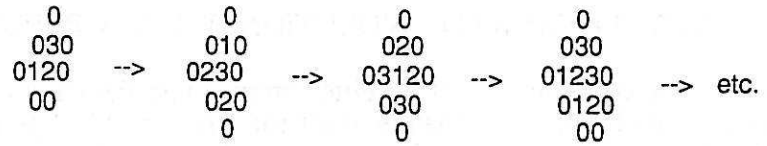


FIG. 9 A FOUR-STATE Cellular Automata REPLICATOR "Arrow"

With hindsight from the previous (staircase) example, a thorough analysis of the transition rules showed that a similar logical impossibility would arise. For example, look at the state "0" cell to the right of the state "3" cell in the cycle 0 configuration. Its state transition rule is 00003-0. Consider now the state "0" cell to the right of the rightmost state "3" cell in the cycle 1 configuration. To attain the target configuration of cycle 2, its state transition rule should be 00003-2, which contradicts the above 00003-0. It was thought that perhaps it was the "corner turning" that was causing the problem. Therefore a configurationally simpler pattern was tried, as shown in FIG. 10

The same weighted fitness definition as above was used for the first two cycles. The resulting chromosome population was used (shaping style) for the third cycle. Those chromosomes not giving a perfect score for the first two cycles during the 3 cycle evolution, scored zero fitness and were thus immediately eliminated from the next generation. The fitness definition of the third cycle evolution was once again :- 100*NCOC + 144 - NNZC, where NCOC was defined for the configuration after the third cycle. FIG. 11 shows the successful cycle by cycle evolution of the elite chromosome. It took 60 generations to evolve the desired cycle 1 configuration, a further 300 generations to evolve the desired cycle 2 configuration, and a further 1000 generations to evolve the desired cycle 3 configuration.

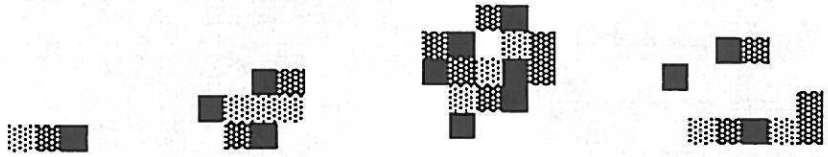
```
 000          0000          00000          000000
01230  -->   023120  -->   0312310  -->  01231230  --> etc.
 000          0000          00000          000000
```

FIG. 10 A FOUR-STATE Cellular Automata REPLICATOR "Line"

Cycle 0 Cycle 1 Cycle 2 Cycle 3

FIG. 11 3 CYCLE EVOLUTION of "LINE" REPLICATOR

FIG. 12 "LINE" REPLICATOR AFTER 18 CYCLES



Cycle 0 Cycle 1 Cycle 2 Cycle 3

FIG. 13 3 CYCLE DIRECT EVOLUTION of "LINE" REPLICATOR

Just to check that the elite chromosome of FIG. 11 had evolved the appropriate state transition rules to achieve continued "line" replication for a greater number of cycles (i.e. > 3), the number of cells in the grid was doubled (to 12*24) and the number of cycles allowed to increase beyond 3. FIG. 12 shows the configuration after 18 cycles. A successful 4 state replicator had been evolved. The 1024 state transition rules of the elite chromosome which generated FIG. 12 are too long to be listed in this paper. Now that it is known that the "line" replicator can be evolved, i.e. that there are no required contradictory state transition rules, a further attempt was made to evolve the line replicator directly, measuring the fitness after the third cycle. This was done because the previous attempts at direct evolution were made impossible by the contradictory state transition rules. FIG. 13 shows the configuration of the elite chromosome at each cycle. The cycle 3 configuration emerged after a few hundred generations but did not change after a further 2000 generations. Thus the direct evolution failed. Having succeeded in evolving a "line" replicator (admittedly with cycle-by-cycle supervised evolution), the next step was to try to evolve a more interesting shape in the same way. FIG. 14 shows the cycle by cycle (0-7) target configurations of a 5 state, 4 cell "square" which replicates every 2 cycles. These squares grow into a large diamond shape. Since the results were identical, FIG. 14 shows also the results obtained from the evolution. The state to gray-level mapping is :-

0-white, 1-light gray, 2-gray, 3-dark gray, 4-black. To ensure that the target configurations of FIG. 14 did not contain contradictory state transition rules, one could if one wanted, handcode the state change rules which generate the cycle T+1 configuration from the cycle T configuration, for all configurations. However, this is a rather pointless exercise because once one has the transition rules, there is no point in evolving them, destroying the very purpose of this paper. Another approach is to initially handcraft several examples in this way, until one develops a "feel" for the type of target patterns which avoid these contradictions. With experience, one soon learns how to avoid the traps, so that one can directly write down contradiction free target configurations. FIG. 14 is a result of such experience.

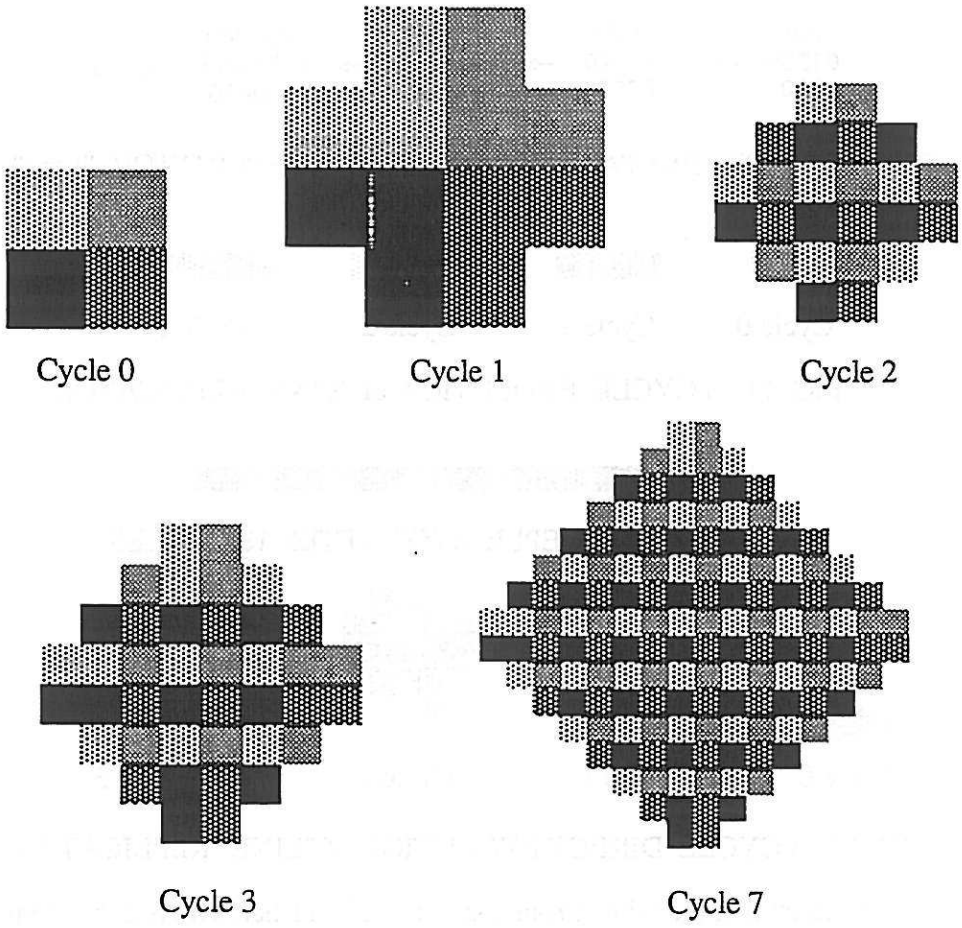


FIG. 14 CYCLES 0-7 of 5-STATE 2D REPLICATOR

To measure the fitness of each configuration, one uses the following formula :-

$$\text{Fitness} = 100*\text{NCPC} + 400 - \text{NNZC}$$

where :-

NCPC	is the Number of Correctly **Positioned** Cells
400 (i.e. 20*20)	is the number of cells in the grid whose states can change
NNZC	is the Number of Non Zero state Cells

This fitness definition differs a little from the earlier definition. Because several new cells appear each cycle, the strict path-like (one cell per cycle) order is not needed. So NCPC, i.e. the number of correctly positioned non zero state cells, is used. However if one is evolving the target configuration for cycle T, then all the actual configurations for cycles < T have to have a perfect score, otherwise the cycle T fitness is immediately made zero, to eliminate the corresponding chromosome from the next generation. This forces the evolution to follow the sequence of target configurations.

The 3125 (i.e. 5^5) state transition rules which were evolved, are not given here for reasons of space. However, the equivalent 76 hand-crafted rules are listed in FIG. 15, to illustrate the point that as the number of hand-crafted rules grows, it gets easier to *evolve* the rules than to handcraft them (provided of course, that the visual consistency checking of the target configurations has been done).

```
00010-1 00020-2 00100-1 00002-2 00400-4 00003-3
04000-4 03000-3 10240-1 20031-2 32004-3 41300-4

00110-3 00022-4 04400-2 03003-1 10210-4 10140-2
11241-1 20021-3 22231-2 20032-1 32334-3 32003-4
33004-2 41400-3 41344-4 44300-1

00130-2 00042-1 00310-4 00024-3 04200-1 03001-2
02400-3 01003-4 14242-1 14002-1 23131-2 23100-2
30420-3 32424-3 41313-4 40013-4

00120-3 00012-4 00240-1 00031-2 00410-3 00023-4
04100-2 03002-1 01300-4 02004-3 03400-2 04003-1
11242-1 10042-1 14241-1 14200-1 20130-2 22131-2
23231-2 23001-2 30024-3 32400-3 32434-3 32324-3
40310-4 41314-4 41343-4 41003-4

00013-4 00420-3 04002-1 03100-2
```

FIG. 15 HAND-CRAFTED TRANSITION RULES FOR FIG. 14

4. Discussion

FIG. 12 above shows that a 4 state cellular automata replicator can be successfully evolved. However, the replicator was relatively simple, so that with a bit of effort, its state transition rules could be found by hand. However the method used in evolving the line replicator can be carried over to replicators with a greater number of states, and having more complex initial and final configurations (e.g. as in FIG. 14). Hand crafting the transition rules of such replicators becomes increasingly tedious. Thus the method developed in this paper to evolve CA replicators "automates" the process of finding state transition rules. The method is probably also amenable to more open ended (less supervised) evolution, a point taken up in more detail later.

The evolution was rather tightly supervised to get it to work. Less supervised evolution failed (e.g. FIG. 13). This was initially rather disappointing, but probably not surprising with hindsight. There are so many evolutionary "blind alleys". The early attempts shown in this paper to evolve configurations which "turned corners" generated contradictory state transition rules. This was unexpected, and showed that even supervised evolution has its pitfalls, because desired target configurations may be logically impossible to obtain from a given initial configuration. Further experiments need to be undertaken to see if more open-ended (less supervised) evolution is possible. For example, it might be interesting to allow the chromosome to code for an initial configuration of cells. The fitness (defined as the degree of partial matching between the original configuration and its "copy") might be measured at each of N cycles, and the chromosome's fitness be defined as the maximum of these cycle fitnesses. This type of approach would eliminate the problem of contradictory state transition rules which can arise with the supervised approach, because impossible targets would not evolve well and hence would get low fitness values.

The CA evolutionary techniques introduced in this paper need not be restricted to replicators. The author has already used Genetic Programming (GP) techniques to evolve artificial embryos using colonies of CA cells to specify "shapes" i.e. contiguous, occupied, 2 state cells [de Garis 1991, 1992, 1993]. Applying GP

methods to CAs may become useful when nanoelectronics starts producing "quantum dot arrays" [Kirk & Reed 1992], where the scale of these next-generation electronic devices is so small that quantum tunneling becomes their essential operating principle. Component densities will be so large, that it will be impossible to connect most components to each other. Very high switching speeds will necessitate short interconnects, i.e. to neighboring components only. The state of each component will then depend upon the states of its neighbors, and thus behave like a cellular automaton. Being able to evolve the configurations of such nanoscale CAs, directly in the hardware, might be very useful for next-generation electronics and computers. It is hoped that the very real difficulties of evolving replicators, as shown by the investigative work of this paper, will challenge those people who wish to evolve nano-scale replicators. Perhaps it might be possible to Genetically Program (GP) molecular replicators in software, using techniques such as molecular mechanics or molecular dynamics (familiar to quantum chemists) to measure the partial fitnesses of potential molecular replicators. The author would very much like to get into "the software evolution of molecular replicators". Whether state-of-the-art super computers can handle such a research topic is an open question. If so, software-based molecular replicator evolution might be preferable to a more expensive wetware-based method, as used for example by Joyce, with his "directed evolution of RNA enzymes" [Joyce 1992]. According to a conversation the author had with Eric Drexler, the state of the art in molecular mechanics is software simulating the interaction of one million atoms.

5. Future Work

The author was rather disappointed by the results. It seems that evolving CA replicators is much harder than initially thought. The replicators evolved in this paper were only achieved under tight supervision, and even then they were rather trivial. The improbabilities of replicator evolution has motivated the author to compromise in future projects (with hand-crafted CA rules and evolved CA "signal sequences"). For example, at the time of writing, the author is engaged in evolving "cellular automata networks" which transmit 4 kinds of signals down their "trails" (e.g. [von Neumann 1966, Codd 1968, Burks 1970, Langton 1984]). When a CA signal hits the end of a CA trail, 4 possible actions can occur, depending on the signal. If the signal is "red" - the trail turns left, if "green" - it turns right, if "brown" - it extends the trail one square, if "purple" - it splits the trail into a T intersection. The sequence of these CA signals (red, green, brown, purple) are treated as chromosomes in a Genetic Algorithm. When two trails collide, a "synapse" is formed which absorbs later signals. The sequence generates a CA network. Once the network is formed, a second set of hand-crafted CA rules makes the net behave like a neural net. For example, signal strengths remain constant in "axons", but decrease as a function of distance from the synapse in "dendrites". These distances from synapses become equivalent to the weights of traditional neural networks. The distances (weights) can be Genetically Programmed. CA "neuron rules" can be hand-crafted, with "CA adders of dendrite signal strengths" etc. The author's program is called CREEPER. If the "evolvability" of these CA-based "neurite networks" proves to be adequate, then these ideas may be put into Cellular Automata Machines (similar in principle to the work of [Margolus & Toffoli 1987]). By having a population of these neurite-network-growing CA Machines, plus some microprocessors to both measure the fitnesses of the neurite networks (at performing some task), and to control the GA, we may build what the author calls a "Darwin Machine" (i.e. a specialised hardware device for Genetic Programming [de Garis 1991, 1993]). One of the longer term aims of the new Evolutionary Systems Department at ATR, is to build artificial brains using perhaps neurite network modules grown on Darwin Machines. Hence the name "Brain Builder Group". This group also includes a "molecular programmer" (i.e. someone who software simulates the design of molecular scale machines and computers), who may attempt to design Darwin Machines at molecular level. Our Evolutionary Systems Department will have a Molecular Computer Architecture Group within a year. It is difficult to imagine how such molecular devices can be built without being genetically programmed.

References

[Burks 1970] "Essays on Cellular Automata", A.W. Burks, University of Illinois Press, 1970.

[Byl 1989] "Self-Reproduction in Small Cellular Automata", J. Byl, Physica D 34, pp 295-299, North Holland, 1989.

[Codd 1968] "Cellular Automata", E.F. Codd, Academic Press, 1968.

[de Garis 1990] "Genetic Programming : Building Artificial Nervous Systems Using Genetically Programmed Neural Network Modules", Hugo de Garis, in Porter B.W. & Mooney R.J. eds., Proc. 7th Int. Conf. Machine Learning, p 132-139, Morgan Kaufmann.

[de Garis 1991] "Genetic Programming : Artificial Nervous Systems, Artificial Embryos and Embryological Electronics", Hugo de Garis, in "Parallel Problem Solving from Nature", Lecture Notes in Computer Science 496, Springer Verlag, 1991.

[de Garis 1992] "Artificial Embryology : The Genetic Programming of an Artificial Embryo", Hugo de Garis, Ch. 14 in book "Dynamic, Genetic and Chaotic Programming", ed. Branko Soucek and the IRIS Group, Wiley, 1992.

[de Garis 1993] "Genetic Programming : GenNets, Artificial Nervous Systems, Artificial Embryos", Hugo de Garis, Wiley manuscript, 1993.

[Drexler 1992] "Nanosystems : Molecular Machinery, Manufacturing and Computation", K.E. Drexler, Wiley, 1992.

[Eigler & Schweizer 1990] "Positioning Single Atoms with a Scanning Tunneling Microscope", D.M. Eigler & E.K. Schweizer, Nature, Vol. 344, 5 April 1990.

[Joyce 1992] "Directed Evolution of an RNA Enzyme", Beaudry A.A. and Joyce G.F., Science, Vol. 257, p 613 & pp 635-641, 31 July 1992.

[Kirk & Reed 1992] "Nanostructures and Mesoscopic Systems", W.P. Kirk & M.A. Reed (eds.), Academic Press, 1992.

[Langton 1984] "Self-Reproduction in Cellular Automata", Physica D, p 135-144, 1984.

[Margolus & Toffoli 1987] "Cellular Automata Machines", N. Margolus & T. Toffoli, Complex Systems 1, pp 967-993, 1987.

[von Neumann 1966] "Theory of Self-Reproducing Automata", J. von Neumann, University of Illinois Press, 1966.

[Wolfram 1986] "Theory & Applications of Cellular Automata", S. Wolfram (ed.), World Scientific, Singapore, 1986.