Artificial Life and Cellular Automata

Robert C. Newman

Introduction

Artificial Life (AL) is a rather new scientific discipline, which didn't really get going until the 1980s.[1] Unlike biology, it seeks to study life not out in nature or in the laboratory, but in the computer.[2] AL seeks to mimic life mathematically, and especially to generate known features of life from basic principles (Langton, 1989b, pp 2-5). Some of the more gung-ho specialists in AL see themselves as creating life in the electronic medium (Ray, 1994, p 180); others think they are only imitating it (Harnad, 1994, pp 544-49). Without addressing this particular question, theists can at least agree that life does not have to be manifested in biochemistry.

Those who believe in metaphysical naturalism C that "the Cosmos is all that is, or ever was, or ever will be" C must presume a purely non-supernatural origin and development of life, unguided by any mind. For metaphysical naturalism, no other kind of causality really exists. Theists, by contrast, believe that a mind C God C is behind it all, however He worked. Perhaps God created matter with built-in capabilities for producing life; perhaps He imposed on matter the information patterns characteristic of living things; perhaps He used some combination of these two.

Current naturalistic explanations of life may generally be characterized by three basic claims. First, that life arose here on earth or elsewhere without any intelligent oversight C a self-reproducing system somehow assembled itself. Second, that the (essentially blind) Darwinian mechanism of mutation and natural selection, which then came into play, was so effective that it produced all the variety and complexity we see in modern life-forms. Third, the time taken for the assembly of the first self-reproducer was short enough, and the rate at which mutation and natural selection operates is fast enough, to account for the general features of the fossil record and such particulars as the "Cambrian explosion." A good deal of AL research seems aimed at establishing one or more of these claims.

What sort of world do we actually live in? The "blind-watchmaker" universe of metaphysical naturalism, or one structured by a designing mind? It is to be hoped that research in AL can provide some input for answering this question.

Meanwhile, the field of AL is already large and is rapidly growing larger. I have neither the expertise nor the space to give a definitive picture of what is happening there. Here we will try to whet your appetite and provide some suggestions for further research by looking briefly at

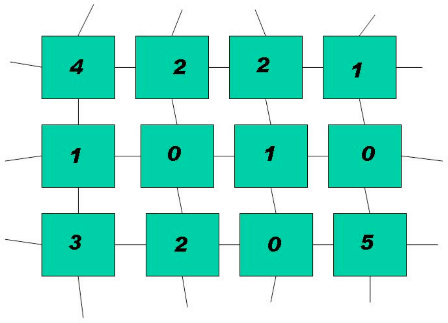
1. See Langton (1989b, pp 6-21) for the "prehistory" of artificial life studies.

2. Taylor and Jefferson (1994, pp 1-4) would define artificial life more broadly, to include synthetic biochemistry and robotics; so too Ray (1994, pp 179-80).

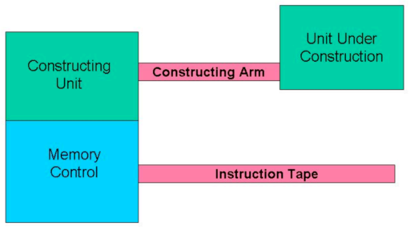
several proposals from AL to see how they are doing in the light of the naturalistic claims mentioned above. First, we shall look at the cellular automata devised by von Neumann, Codd, Langton, Byl and Ludwig, both as regards the origin of significant self-reproduction and the question of how life might develop from these. Second, we will sketch Mark Ludwig's work on computer viruses, which he suggests are the nearest thing to artificial life that humans have yet devised. Third, we will examine one of Richard Dawkins' programs designed to simulate natural selection. Fourth, we will look at Thomas Ray's "Tierra" environment, which seeks to explore the effects of mutation and natural selection on a population of electronic creatures.

Cellular Automata

Beginning nearly half a century ago, long before there was any discipline called AL, computer pioneer John von Neumann sought to investigate the question of life's origin by trying to design a self-reproducing automaton. This machine was to operate in a very simplified environment to see just what was involved in reproduction. For the building blocks of this automaton, von Neumann decided on computer chips fixed in a rigid two-dimensional array rather than biochemicals swimming in a three-dimensional soup. [In practice, his machine was to be emulated by a single large computer to do the work of the many small computer chips.]



Each computer chip is identical, but can be made to behave differently depending on which of several operational states it is currently in. Typically we imagine the chips as wired to their four nearest neighbors, each chip identifying its current state via a number on a liquid crystal display like that on a wristwatch. The chips change states synchonously in discrete time-steps rather than continuously. The state of each chip for the next time-step is determined from its own current state and those of its four neighbors using a set of transition rules specified by the automaton's designer.



The idea in the design of a self-reproducing automaton is to set up an initial array of states for some group of these chips in such a way that they will turn a neighboring set of chips into an information channel, and then use this channel to "build" a copy of the original array nearby.

Von Neumann in the late '40s and early '50s attempted to design such a system (called a cellular automaton) that could construct any automaton from the proper set of encoded instructions, so that it would make a copy of itself as a special case. But he died in 1957 before he could complete his design, and it was finished by his associate Arthur Burks (von Neumann, 1966). Because of its complexity C some 300x500 chips for the memory control unit, about the same for the constructing unit, and an instruction "tape" of some 150,000 chips C the machine von Neumann designed was not built.

Since von Neumann's time, self-reproducing automata have been greatly simplified. E. F. Codd (1968) reduced the number of states needed for each chip from 29 to 8. But Codd's automaton was also a "universal constructor" C able to reproduce any cellular automaton including itself. As a result, it was still about as complicated as a computer.

Christopher Langton (1984) made the real break-through to simplicity by modifying one of the component parts of Codd's automaton and from it producing a really simple automaton (shown below) that will reproduce itself in 151 time-steps. It reproduces by extending its arm (bottom right) by six units, turning left, extending it six more units, turning left, extending six more, turning left a third time, extending six more, colliding with the arm near its beginning, breaking the connection between mother and daughter, and then making a new arm for each of the two automata. Langton's automaton, by design, will not construct other kinds of cellular automata as von Neumann's and Codd's would. His device consisted of some 10x15 chips, including an instruction tape of 33 chips, plus some 190 transition rules.

```
 22222222
2170140142
2022222202
272    212
212    212
202    212
272    212
21222222122222
207107107111112
 2222222222222
```

Just a few years later, John Byl (1989a, b) simplified Langton's automaton further (see below) with an even smaller automaton that reproduced in just 25 time-steps. Byl's automaton consisted of an array of 12 chips C of which 4 or 5 could be counted as the instruction tape C and 43 transition rules.

3. A forthcoming article (Pesavento, 1995) announces the recent implementation of von Neumann's machine.

```
T = 0    T = 5    T = 10

 22       22       22  3

2632     2342     2462

2642     266633   23664363

 25       2212     221222
```

```
T = 15     T = 20       T = 25

 22  3       22     2     22   22

2662   3    2632  362   2345 2632

243664362   264366432   2662 2642

 2212222     2212222     22 2 25
```

Most recently, Mark Ludwig (1993, pp. 107-108) has apparently carried this simplification to its limit with a miniscule automaton that reproduces in just 5 time-steps. This automaton consists of 4 chips, only one of which is the instruction "tape," and some 22 transition rules.

```
T = 0   T = 1   T = 2   T = 3   T = 4   T = 5


  2       2       2       2       2       25

 212     212     212     21      213     21 4

          3        5      2       2       2

                  4      636     626      2

                          6      262     212
```

It is interesting to note that the information contained in each of these self-reproducing automata may be divided into three parts: (1) the transition rules, (2) the geometry of the chips, and (3) the instruction tape. (1) The transition rules, which tell us how state succeeds state in each chip, somewhat resemble the physics or chemistry of the environment in the biological analogue. (2) The geometry of the automaton would correspond to the structure of a biological cell. (3) The instructions resemble the DNA. Thus these automata have a division of information which corresponds to that found in life as we know it on earth. In both cases self-reproduction depends not only on an instruction set, but also upon the structure of the reproducer and the nature of the physical realm in which it operates.

For the von Neumann and Codd automata, since they are universal constructors, the size of the machine and its instructions are enormous! One could not seriously entertain a naturalistic origin of life if the original self-reproducing system had to have anything like this complexity.

The smaller automata look much more promising, however. Perhaps a self-reproducing biochemical system at this level of complexity could have arisen by a chance assembly of parts. In a previous paper (Newman, 1988) I suggested that the random formation of something as complex as the Langton automaton (even with very generous assumptions) was out of the question in our whole universe in the 20 billion years since the big bang, as the probability of formation with all this space and time available is only 1 chance in 10^{129}.

In response to Byl's proposed automaton, I found it necessary (Newman, 1990a) to retract some of the generosity given to Langton, but by doing so found that even Byl's automaton had only 1 chance in 10^{69} of forming anywhere in our universe since the big bang.

Ludwig's automaton looks so simple as to be a sure thing in a universe as vast and old as ours is. Indeed, by the assumptions used in doing my probability calculation for Byl's automaton, we would have a Ludwig automaton formed every 7×10^{-15} seconds in our universe.

However, an enormously favorable assumption is contained in this calculation C that all the carbon in the universe is tied up in 92-atom molecules which exchange material to try out new combinations as quickly as an atom can move the length of a molecule at room temperature. If, however, we calculate the expected fraction of carbon that would actually be found in 92-atom polymers throughout our universe, the expected time between formation of Ludwig automatons in our universe jumps to about 10^{86} years! Thus it would still not be wise to put one's money on the random formation of self-reproduction even at this simple level.

Besides the problem of formation time, the physics (transition rules) of these smaller automata was specially contrived to make the particular automaton work, and it is probably not good for anything else. Since the automata of Langton, Byl and Ludwig were not designed to be universal constructors, self-reproduction typically collapses for any mutation in the instructions. To avoid this, the constructing mechanism in any practical candidate for the first self-reproducer will have to be much more flexible so that it can continue to construct copies of itself while it changes.

The physics of such automata could be made more general by going back toward the larger number of states used in von Neumann's automaton. Langton, for instance, has a signal for extending a data path in his information tape, but none for retracting one; a signal for a left-turn, but none for a right-turn. These could be included rather easily by adding additional chip states to his eight, thus making the physics more flexible. Of course this would significantly increase the number of transition rules and the consequent complexity of his automaton.

This, obviously, makes self-reproduction even less likely to have happened by chance.

But it would also help alleviate the problem that these simpler automata don't have a big enough vocabulary in their genetic information systems to be able to do anything but a very specialized form of self-reproduction, and they have no way to expand this vocabulary which was designed in at the beginning. This problem seems to me a serious one for the evolution of growing levels of complexity in general.

As for building an automaton that is more general in its constructing abilities and not tied to a particular physics especially contrived for it, Karl Sigmund (1993, pp. 27-39) has described an attempt by John Conway to use the environment of his game of "Life" as a substrate on which to design a universal constructor. He succeeds in doing so, but the result is outrageously complex, back in the league with von Neumann's and Codd's automata.

We should be able to design a *somewhat* general self-reproducing automaton on a substrate not especially designed for it. This would be a good project for future research. We would then have a better handle on what complexity appears to be minimal for significant self-reproduction, and what would be the likelihood it could occur by chance in a universe such as ours.

The environment in which each of these self-reproducing automata operates is empty in the sense that nothing else is around and happening. By design, the sea of unused cells is quiescent. This is certainly unlike the scenario imagined for the origin of biochemical life. What will happen to our automata if they are bumped by or run into other objects in their space? Are they too fragile to be real candidates for the hypothetical original self-reproducer? The Langton automaton certainly is. By running the program with a "pimple" placed on the surface of the automaton (i.e., the structure is touched by a single cell in any of the states 1-7), we find that the automaton typically "crashes" in about 30 time-steps (the time taken for the data to cycle once around the loop). It appears that the automaton is very fragile or "brittle" rather than robust. This would certainly not be satisfactory in a real-life situation.

Computer Viruses

Mark Ludwig, PhD in elementary particle physics, proprietor of American Eagle Publications, and author of *The Little Black Book of Computer Viruses* (1991), has written a very stimulating book entitled *Computer Viruses, Artificial Life and Evolution* (1993). Ludwig argues that computer viruses are really much closer to artificial life than anything else humans have produced so far, especially in view of the fact that such viruses have gotten loose from their creators (or been set loose) and, like the biochemical viruses for which they are named, are fending for themselves rather successfully in a hostile environment.

Like the various cellular automata we discussed above, computer viruses have the ability to reproduce themselves. In addition, they can typically hide themselves from "predators" (anti-

4. See also Spafford (1994).

virus programs) by lurking inside the instructions of some regular computer program which they have "infected." They may also function as parasites, predators, or just clever annoyances as they ride programs from disk to disk, computer to computer, and user to user. Some viruses (by design or not) damage files or programs in a computer's memory; others just clutter up memory or diskettes, or send humorous and irksome messages to the computer screen.

So far as I know, no one claims that computer viruses arose spontaneously in the memories of computers. But how likely would it be for something as complex as a simple virus to form by chance in the computer environment?

Early in 1993, Ludwig sponsored "The First International Virus Writing Contest," awarding a prize for the shortest virus that could be designed having certain rather minimal function (Ludwig, 1993, pp 319-321). He provides the code (computer program) for the virus that was grand prize winner and for several runners-up, plus a sample virus which he sent out with the original announcement of the contest (Ludwig, 1993, pp 322-331). These programs all turned out to be over 100 bytes in length.

Ludwig calculates for the shortest of these (101 bytes) that there are 10^{243} possible files of length 101 bytes. If we could get all the 100 million PC users in the world to run their machines full-time with a program that generates nothing but 101-byte random sequences at 1000 files per second, then in 10 years the probability of generating this particular virus is 2×10^{-224} (ibid., p 254-5). If they ran for the whole history of the universe, the probability would be 4×10^{-214}. If all the elementary particles in our universe were converted into PCs generating 1000 random 101-byte files per second, the probabily of forming this particular virus would be 6×10^{-110} (ibid., p 255). Obviously our universe does not have the probabilistic resources to generate this level of order by random assembly!

Ludwig then discusses two much smaller programs. One is a rather crude virus of 42 bytes, which just copies itself on top of all the programs in a computer's directory. He notes that one might just expect to form this virus in the history of the universe if all those elementary particles were PCs cranking out 1000 42-byte random files per second, but that if one only had the 100 million PCs and ten years for the job, the probability would be only 4×10^{-81} (ibid., pp 254-5). This would improve to 8×10^{-71} if one had the time since the big bang to work with.

The smallest program Ludwig works with is not a virus, since it cannot make copies of itself that are saved to disk, but only copies that remain in memory so long as the computer is running. This program is only 7 bytes long. It could easily be formed in ten years with 100 million PCs turning out 1000 7-byte sequences per second, but it would take a single computer about 2.5 million years to do so.

It is doubtful that this is the long-sought self-reproducer that will show life arose by chance. The actual complexity of this program is considerably greater than 7 bytes because it uses the copying routine provided by the computer. The environment provided for computer viruses is much more helpful for self-reproduction than is the biochemical environment.

As in the case of cellular automata, we see that a random search for self-reproduction (before mutation and natural selection can kick in) is an extremely inefficient way to reach even very modest levels of organized complexity; but for naturalism, that is the only path available.

Ludwig also considers forming a virus by accidental mutation of an existing computer program (Ludwig, 1993, pp 259-263). This is an interesting discussion, but it tells us more about how a biochemical virus might get started in a world which already has a lot of life than it does about how life might get started in abiotic circumstances.

Dawkins' "Weasel" Program

Richard Dawkins claims that there is no need for a mind behind the universe. Random processes, operating long enough, will eventually produce any level of order desired. "Give enough monkeys enough time, and they will eventually type out the works of Shakespeare."

If indeed we grant that we live in a universe totally devoid of mind, then something like this *must* be true. And granting this, if we broaden our definition of "monkey" sufficiently to include anthropoid apes, then it has *already happened*! An ape evolved into William Shakespeare who eventually wrote C and his descendants typed C his immortal works!

But seriously, this is merely to beg the question. As Dawkins points out (Dawkins, 1987, pp 46-47), the time required to reasonably expect a monkey to type even one line from Shakespeare C say "Methinks it is like a weasel" from *Hamlet* C would be astronomical. To get any significant level of order by random assembly of gibberish is out of the question in a universe merely billions of years old and a similar number of light-years across.

But Dawkins (who, after all, believes our universe was devoid of mind until mind evolved) claims that selection can vastly shorten the time necessary to produce such order. He programs his computer to start with a line of gibberish the same length as the target sentence above and shows how the target may be reached by selection in a very short time.

Dawkins accomplishes this (ibid., pp 46-50) by having the computer make a random change in the original gibberish and test it against the target sentence, selecting the closer approximation at each step and then starting the next step with the selected line. For instance, starting with the line:

WDLTMNLT DTJBSWIRZREZLMQCO P

Dawkins' computer reaches its target in just 43 steps or "generations." In two other runs starting with different gibberish, the same target is reached in 64 and 41 generations.

This is impressive C but it doesn't tell us much about natural selection. A minor problem with Dawkins' program is that he has designed it to converge far more rapidly than real mutation and selection would. I devised a program SHAKES (Newman, 1990b) which allows the operator to enter any target sentence plus a line of gibberish of the same length. The computer then randomly chooses any one of the characters in the line of gibberish, randomly chooses what change to make in that character, and then tests the result against the target. If the changed line is closer to the target than it was before the change, it replaces the previous gibberish. If not, then the previous version remains. Dawkins did something like this, but his version closes on its target far more rapidly. For instance his version moves from

METHINKS IT IS LIKE I WEASEL

to

METHINKS IT IS LIKE A WEASEL

in just three generations (Dawkins, 1987, p 48). I suspect that what Dawkins has done is that once the computer gets a particular character right, it never allows mutation to work on that character again. That is certainly not how mutation works! My version took several hundred steps to move across a gap like the one above because the mutation both had to randomly occur at the right spot in the line and randomly find a closer letter to put in that place. My runs typically took over a thousand steps to converge on the target from the original gibberish.

But a far more serious problem with Dawkins' simulation is that real mutation and natural selection don't have a template to aim at unless we live in a designed universe (see Ludwig, 1993, pp 256-259). A better simulation would be an open-ended search for an unspecified but meaningful sentence, something like my program MUNSEL (Newman, 1990b). This program makes random changes in the length and the characters of a string of letters without a template guiding it to some predetermined result. Here a randomizing function either adds a letter or space to one end of the string, or changes one of the existing letters or spaces to another. This is intended to emulate the action of mutation in changing the nucleotide bases in a DNA molecule or the amino acids in a protein.

In this program, natural selection is simulated by having the operator manually respond as to whether or not the resulting string consists of nothing but English words. If it does, then the mutant survives (is retained); if it doesn't, the mutant dies (is discarded). This could be done more efficiently (and allow for much longer computer runs) if one would program the computer to use a spell-checker from a word-processing program to make these decisions instead of a human operator.

Even more stringent requirements might be laid on the mutants to simulate the development of higher levels of order. For instance, the operator might specify that each successful mutant conform to English syntax, and then that it make sense on larger and larger size-scales. This would give us a better idea of what mutation and natural selection can do in producing such higher levels of organization as would be necessary if macroevolution is really to work.

Ray's "Tierra" Environment

One of the most interesting and impressive attempts at the computer simulation of evolution I have seen so far is the ongoing experiment called "Tierra," constructed by Thomas Ray at the University of Delaware (Ray, 1991). Ray designed an electronic organism that is a small computer program which copies itself. In this it resembles cellular automata and particularly computer viruses. It differs from these in that it lives in an environment C "the soup," also designed by Ray C which explicitly includes both mutation and a natural competition between organisms.

To avoid problems that can arise when computer viruses escape captivity, the soup is a "virtual computer," a text file that simulates a computer, so the programs are not actually roaming around loose in the computer's memory. For most of Ray's runs, the soup contains 60,000 bytes, equivalent to 60,000 instructions. This will typically accomodate a population of a few hundred organisms, so the dynamics will be those of a small, isolated population.

To counter the problem of fragility or brittleness mentioned in our discussion of cellular automata, Ray invented his own computer language. This "Tierran" language is more robust than the standard languages, so not as easily disrupted by mutations. It is a modification of the very low-level assembly language used by programmers, with two major differences: (1) it has very few commands C only 32 (compare the assembly language for 486 computers, with nearly 250 commands [Brumm, 1991, 136-141]) C and (2) it addresses other locations in memory by the use of templates, rather than address numbers C a feature modelled on the biochemical technique by which molecules "find" each other. The program is set up so the operator can vary the maximum distance that an organism will search to locate a needed template.

Ray starts things off by introducing a single organism into the soup. There it begins to multiply, with the mother and resulting daughter organisms taking turns at copying themselves until they have nearly filled the available memory. Once the level of fullness passes 80%, a procedure kicks in which Ray calls "the Reaper." This keeps the soup from overcrowding by killing off organisms one-by-one, working down from the top of a hit list. An organism at birth starts at the bottom of this list and moves upward as it ages, but will move up even faster if it makes certain errors in copying. Alternatively, it can delay moving upward somewhat if it can successfully negotiate a couple of difficult procedures.

The master computer which runs the simulation allows each organism to execute its own instructions in turn. The turn for each organism can be varied in different runs of the experiment so as to make this allowance some fixed number of instructions per turn, or dependent on the size of the organism so as to favor larger creatures, smaller ones, or be size-neutral.

Ray introduces mutation into the system by fiat, and can change the rate of mutation from zero (to simulate ecological situations on a timescale much shorter than the mutation rate) up to very high levels (in which the whole population perishes).

One form of mutation is designed to simulate that from cosmic rays. Binary digits are flipped at random locations in the soup, most of which will be in the organisms' genomes. The usual rate which Ray sets for this is one mutation for every 10,000 instructions executed.

Another form of mutation is introduced into the copying procedure. Here a bit is randomly flipped during reproduction (typically for every 1000 to 2500 instructions transfered from mother to daughter). This rate is of similar magnitude to the cosmic ray mutation.

A third source of mutation Ray introduces is a small level of error in the execution of instructions, making their action slightly probabilistic rather than strictly deterministic. This is intended to simulate occasional undesired reactions in the biochemistry (Ray, 1994, p 187). Ray does not specify the rate at which error in introduced by this channel.

Ray's starting organism consists of 80 intructions in the Tierran language, each instruction being one byte (of 5 bits) long. The organism begins its reproduction cycle by reading and recording its length, using templates which mark the beginning and end of its instruction set. It then allocates a space in the soup for its daughter, and copies its own instructions into the allocated space, using other templates among its instructions for the needed jumps from place to place in its program (subroutines, loops, etc.). It ends its cycle by constituting the daughter a separate organism. Because the copying procedure is a loop, the original unmutated organism actually needs to execute over 800 instructions before it completes one full reproduction. Once there are a number of organisms in the soup, this may require an organism to use several of its turns to complete one reproduction.

Ray has now run this experiment on his own personal computer and on much faster mainframe computers many times, with some runs going for billions of instructions. (With 300 organisms in the soup, 1 billion instructions would typically correspond to some four thousand generations.) Ray has seen organisms both much larger and much smaller than the original develop by mutation, and some of these have survived to do very well in the competition.

Ray has observed the production of parasites, which have lost the instructions for copying themselves, usually due to a mutation in a template that renders it useless. These are sterile in isolation, but in the soup they can often use the copy procedure of a neighbor by finding its template. This sort of mutant typically arises in the first few million instructions executed in a run (less than 100 generations after the soup fills). Longer runs have produced (1) organisms with some resistance to parasites; (2) hyper-parasites, which cause certain parasites to reproduce the hyper-parasite rather than themselves; (3) social hyper-parasites, which can reproduce only in communities; and (4) cheaters, that take advantage of the social hyper-parasites. All these Ray would classify as microevolution.

Under the category of macroevolution, Ray mentions one run with selection designed to favor large-sized organisms, which produced apparently open-ended size increase and some organisms longer than 23 thousand instructions.

Ray notes two striking examples of novelty produced in his Tierra simulations: (1) an unusual procedure one organism uses to measure its size, and (2) a more efficient copying technique developed in another organism by the end of a 15-billion-instruction run. In the former of these, the organism, having lost its template that locates one end of its instructions, makes do by using a template located in the middle and multiplying this length by two to get the correct length. In the latter, the copying loop has become more efficient by copying three instructions per loop instead of just one, saving the execution of several steps.

With size-neutral selection, Ray has found periods of stasis punctuated by periods of rapid change. Typically, the soup is first dominated by organisms with length in the order of 80 bytes for the first 1.5 billion instructions executed. Then it comes to be dominated by organisms 5 to 10 times larger in just a few million more instructions. In general it is common for the soup to be dominated by one or two size-classes for long periods of time. Inevitably, however, that will break down into a period (often chaotic) in which no size dominates and sometimes no genotypes are breeding true. This is followed by another period of stasis with one or two other size classes now dominating.

Ray's results are impressive. But what do they mean? For the origin of life, not much. Ray has not attempted to simulate the origin of life, and his creatures at 80 bytes in length are complex enough to be very unlikely to form by chance. Each byte in Tierran has 5 bits or 32 combinations, so there are 32^{80} combinations for an 80-byte program, which is 2×10^{120}. Following Ludwig's scheme of using all the earth's 100 million PCs to generate 1000 80-byte combinations per second, we would need 7×10^{100} years for the job. If all 10^{90} elementary particles were turned into computers to generate combinations, it would still take 7×10^{10} years, several times the age of the universe. Not a likely scenario, but one might hope a shorter program that could permanently start reproduction might kick in much earlier.

What about the type of evolution experienced in the Tierra environment? Is it such that we would expect to reach the levels of complexity seen in modern life in the available timespan? It is not easy to answer this. The Tierra simulation is typically run with a very high rate of mutation, perhaps on the order of 1 in 5000 counting all three sources of mutation. Copying errors in DNA are more like 1 in a billion (Dawkins, 1987, p. 124), some 200,000 times smaller. Thus we get a lot more variation in a short time and many more mutations per generation per instruction. Ray justifies this by claiming that he is emulating the hypothetical RNA world before the development of the more sophisticated DNA style of reproduction, and that a much higher level of mutation is to be expected. Besides, for the sake of simulation, you want to have something to study within the span of reasonable computer times. All this is true, but there is also the danger of simulating a world that is far more hospitable to evolution than ours is (see the remark of Pattee and Ludwig's discussion in Ludwig, 1993, pp. 162-164).

The consequences of mutation seem considerably less drastic in Tierra also, making that world especially favorable for evolution. No organism in Tierra dies before it gets a shot at reproducing, whereas dysfunction, disease, predators and accidents knock off lots of these (fit or not) before they reproduce in our world. This effectively raises the mutation rate in Tierra still higher while protecting against some of its dangers, and increases the chance that an organism may be able to hop over a gap of dysfunction to land on an island of function.

In Tierra, the debris from killed organisms remains in the environment. But instead of being a danger to living organisms as it so often is in our world, the debris is available as instructions for parasites whose programs are searching the soup for templates. This enormously raises the mutation rate for parasites, producing something rather like sexual reproduction in a high mutation environment.

Tierran organisms have rather easy access to the innards of other organisms. The program design allows them to examine and read the instructions of their neighbors, but not write over them. The organisms are designed to be able to search in either direction from their location some distance to find a needed template. This is typically set at 200-400 instructions, but on some runs has been as high as 10,000, giving access to one-third the entire environment! This feature is not used by any organism whose original templates are intact, but it provides the various types of parasites with the opportunity to borrow genetic material from up to one-third of the creatures in Tierra, and probably permits many of them to escape their parasitic lifestyle with a whole new set of genes.

The innards themselves, whether part of a living or dead organism, are all nicely usable instructions. Every byte in each organism is an instruction, and once an organism has inhabited a particular portion of the soup, its instructions are left behind after its death until written over by the instructions of another organism inhabiting that space at a later time.

The Tierran language is very robust; every mutation of every byte produces a mutant byte which makes sense within the system. Gibberish only arises in the random arrangement of these bytes rather than in any of the bytes themselves. Thus, the Tierran language cannot help but have meaning at the level of words. The real test, then, for macroevolution in Tierra will be how successful it is in producing meaning at the level of sentences, and this does not appear impressive so far.

There is a need for someone with facility in reading assembly language to take a look at the Tierran mutants to see what evolved programs look like. How do these compare with the programs found in the DNA of biochemical life? Are they comparable in efficiency, in elegance and in function? Does the DNA in our world look as though biochemical life has followed a similar history to that of these Tierran creatures?

Thomas Ray's experiment needs to be continued, as it is a sophisticated procedure for demonstrating what an evolutionary process can actually accomplish. But the details of its design need to be continually revised to make it more and more like the biochemical situation.

5. Some helpful attempts in this direction have been made by Adami and Brown (1994) and Shanahan (1994).

Ray has shown that the Tierra environment can occasionally produce apparent design by accident. Can it produce enough of this to explain the proliferation and sophistication of apparent design we actually have in biochemical life on earth?

In a more recent paper, Ray has begun an attempt to mimic multicellular life (Thearling and Ray, 1994). So far, they have been unable to produce organisms in which the cells are differentiated. And they have skipped the whole problem of how to get from unicellular to multicellular life.

One might wish to say that the Tierran language is too restricted to be able to accomplish all the things that have happened in the history of life on earth. But Maley (1994) has shown that the Tierran language is computationally complete C that it is equivalent to a Turing machine, so that in principle it can accomodate any function that the most sophisticate computer can perform. Of course, it might take an astronomically longer time to accomplish this than a really good computer would, but that brings us back to the question of whether simulations might be more efficient or less efficient than biochemistry to produce the sort of organization we actually find in nature. Until we can answer this, it will be hard to use AL to prove that life in all its complexity could or could not have arisen in our universe in the time available.

Conclusions

We've made a rather rapid (and incomplete) tour of some of the things that are happening in Artificial Life research. The field is growing and changing rapidly, but we should have a better handle in just a few years on the questions of how complex self-reproduction is and what random mutation and natural selection are capable of accomplishing. At the moment, things don't look too good for the "Blind Watchmaker" side.

The definition of self-reproduction is somewhat vague, and can be made much too easy (compared to the biochemical situation) in some computer simulations by riding on the copying capabilities of the host computer and its language. We need to model something that is much more similar to biochemistry.

A self-reproducing automaton apparently needs to be much closer to a universal constructor than the simplest self-reproducers that have been proposed. In order that it not immediately collapse when subjected to any mutation, it must be far more robust. It must be able to continue to construct itself as it changes from simple to complex. In fact, it must somehow change both itself and its instructions in synchronism in order to survive (continue to reproduce) and develop the levels of complexity seen in biochemical life. This is a tall order indeed for any self-reproducer that could be expected to form in a universe as young and as small as ours is. Of course, it can certainly be done if we have an infinite number of universes and an infinite time-span to work with, but there is no evidence that points in this direction.

6. See Dewdney (1989) for a discussion of Turing machines.

In biochemical life, multicellular organisms have a rather different way of functioning than do single cell creatures, and a very different way of reproducing. Clearly, some real change in technique is introduced at the point of transition from one to the other, that is, at the Cambrian explosion. So far, nothing we have seen in computer simulations of evolution looks like it is capable of the things that happened then.

At the moment, AL looks more like an argument for design in nature than for a universe without it.

Bibliography

Ackley, D. H. and Littman, M. L.: 1994. A Case for Lamarckian Evolution. In Langton, 1994, pp 3-9.

Adami, C. and Brown, C. T.: 1994. Evolutionary Learning in the 2D Artificial Life System "Avida." In Brooks and Maes, 1994, pp 377-381.

Adami, C.: 1995. On Modeling Life. *Artificial Life* 1:429-438.

Agarwal, P.: 1995. The Cell Programming Language. *Artificial Life* 2:37-77.

Bonabeau, E. W. and Theraulaz, G.: 1994. Why Do We Need Artificial Life? *Artificial Life* 1:303-325.

Brooks, R. A. and Maes, P.: 1994. *Artificial Life IV*. Cambridge, MA: MIT Press.

Brumm, P., Brumm, D., and Scanlon, L. J.: 1991. *80486 Programming*. Blue Ridge Summit, PA: Windcrest.

Byl, J.: 1989a. Self-Reproduction in Small Cellular Automata. *Physica D*, 34:295-299.

Byl, J.: 1989b. On Cellular Automata and the Origin of Life. *Perspectives on Science and Christian Faith*, 41(1):26-29.

Codd, E. F.: 1968. *Cellular Automata*. New York: Academic.

Dawkins, R.: 1987. *The Blind Watchmaker*. New York: Norton.

Dennett, D.: 1994. Artificial Life as Philosophy. *Artificial Life* 1:291-292.

Dewdney, A. K.: 1989. *The Turing Omnibus*. Rockville, MD: Computer Science Press.

Fontana, W., et al: 1994. Beyond Digital Naturalism. *Artificial Life* 1:211-227.

Harnad, S.: 1994. Artificial Life: Synthetic vs. Virtual. In Langton, 1994, pp 539-552.

Koza, J. H.: 1994. Artificial Life: Spontaneous Emergence of Self-Replicating and Evolutionary Self-Improving Computer Programs. In Langton, 1994, pp 225-262.

Langton, C. G.: 1984. Self-Reproduction in Cellular Automata. *Physica D*, 10:135-144.

Langton, C. G. (ed.): 1989a. *Artificial Life*. Reading, MA: Addison-Wesley.

Langton, C. G.: 1989b. Artificial Life. In Langton, 1989a, pp 1-47.

Langton, C., Taylor, C., Farmer, D., and Rasmussen, S. (eds.): 1991. *Artificial Life II*. Redwood City, CA: Addison-Wesley.

Langton, C. G. (ed.): 1994. *Artificial Life III*. Reading, MA: Addison-Wesley.

Ludwig, M.: 1991. *The Little Black Book of Computer Viruses*. Tucson, AZ: American Eagle Publications.

Ludwig, M.: 1993. *Computer Viruses, Artificial Life and Evolution*. Tucson, AZ: American Eagle Publications. Current address: American Eagle Publications, PO Box 1507, Show Low, AZ 85901.

Maley, C. C.: 1994. The Computational Completeness of Ray's Tierran Assembly Language. In Langton, 1994, pp 503-514.

Neumann, J. von: 1966. *The Theory of Self-Reproducing Automata*, ed. A. W. Burks. Urbana, IL: University of Illinois.

Newman, R. C.: 1988. Self-Reproducing Automata and the Origin of Life. *Perspectives on Science and Christian Faith*, 40(1):24-31.

Newman, R. C.: 1990a. Automata and the Origin of Life: Once Again. *Perspectives on Science and Christian Faith*, 42(2):113-114.

Newman, R. C.: 1990b. *Computer Simulations of Evolution*. MS-DOS diskette of computer programs. Hatfield, PA: Interdisciplinary Biblical Research Institute.

Pesvento, U.: 1995. An Implementation of von Neumann's Self-Reproducing Machine. *Artificial Life* 2(4). In press.

Ray, T. S.: 1991. An Approach to the Synthesis of Life. *Artificial Life II* ed. C. Langton, C. Taylor, J. D. Farmer and S. Rasmussen, pp 371-408. Redwood City, CA: Addison-Wesley.

Ray, T. S.: 1994. An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life. *Artificial Life* 1:179-209.

Shanahan, M.: 1994. Evolutionary Automata. In Brooks and Maes, 1994, pp 388-393.

Sigmund, K.: 1993. *Games of Life: Explorations in Ecology, Evolution, and Behaviour*. New York: Oxford.

Sipper, M.: 1995. Studying Artificial Life Using a Simple, General Cellular Model. *Artificial Life* 2:1-35.

Spafford, E. H.: 1994. Computer Viruses as Artificial Life. *Artificial Life* 1:249-265.

Taylor, C. and Jefferson, D.: 1994. Artificial Life as a Tool for Biological Inquiry. *Artificial Life* 1:1-13.

Thearling, K. and Ray, T. S.: 1994. Evolving Multi-Cellular Artificial Life. In Brooks and Maes, 1994, pp 283-288.