Origins of Evolution

Michael Lachman

The Interdisciplinary Program
for Fostering Excellence
Tel-Aviv University
Tel-Aviv, 69978, Israel

May 12, 1991

Abstract

It is claimed that the subject of the origin of life and evolution should be studied by using general laws, without regard to the chemical or physical constraints on primordial earth. A simulation is introduced with which the origin of evolution was studied. Three typical behaviors which emerged in the system are shown: Self-reproduction, unchanging freezing, and chaotic growth. These behaviors are discussed and their implication on the definition of Evolution is examined.

1 Introduction

Theories of Biological evolution regard the laws of physics and chemistry as the constraints imposed on the evolving system. The guiding principles of Biological evolution — variation, heredity and multiplication — are in a sense independent of these constraints. It is clear that the above general principles can be applied to any system, whatever its nature (chemical, physical or other), and this system will "evolve".

In the study of the origin of life, the theoretical models are constructed to simulate the specific chemical constraints on the primordial

earth. I would like to study the origin of life and Evolution by asking a more general question: In what kinds of systems, regardless of their nature, does evolution emerge? In other words, in what kinds of dynamical systems does Evolution appear? To answer this question we have to define the terms "Evolution" and "appear" or "emerge". Some preliminary attempts to define Evolution will be made in the following sections. The definition of the notion of existence will not be discussed in the present paper.

In order to understand Evolution I shall consider the following points:

First various dynamical systems will be examined. Looking for phenomena in these systems which can be identified as "Evolution".s If a distinct phenomenon is found, some insight into the notion of Evolution will be gained.

Then we will be able to define mathematically a notion of Evolution. After this it will be possible to prove some of the conditions necessary and/or sufficient in a dynamical system for Evolution to exist in it.

2 A simulation for studying Evolution

This section presents a computer simulation for studying Evolution. This is one of the first kinds of simulations atempted, and its main function is to gain some insight into the circumstances in which Evolution can begin. This should enable better simulations to be made in the future.

The following principles were used to build the laws of this simulation:

1. Evolution should be possible. This means that the laws should be rich enough for the universe to house Evolution. The only example of Evolution we have (biological Evolution) works using self-replicating forms, so the laws were built so that self-replicating forms should be possible.
2. Evolution should rise easily. For instance one could have chosen the rules of the cellular-automatum 'LIFE', but in this model

self-replicating forms (universal constructors) are of a size of approximately $10^6 \cdot 10^6$ bits, and thus one would need a very large computer to house Evolution.

3. The laws should act on a lower level than the level with which Evolution deals. For instance, the laws should not include a particular mutation-rate, or any definition of "fitness". These will emerge in the process of the appearance of Evolution. Had we simulated the laws of physics in a computer to study Evolution (which we do not) the laws would act on a lower level than the laws of evolution. Whereas if we biological evolution by simulating natural selection, (as is done in most models of population genetics) we would have to define the fitness, mutation rate etc. as parameters of the simulation externally.

4. The laws can be chosen quite arbitrarily. When looking for a common phenomenon likely to emerge in many systems, an arbitrary system can be chosen for inspection, just as when checking a formula, some insight might be gained by testing it on arbitrary numbers.

2.1 The laws of the universe

2.1.1 The system

The system consists of a number of cells (memory cells), which are ordered sequentially and numbered 0–7999. Each such cell contains 2 numbers:

1. The type of instruction (1–46)
2. The amount of food [1] the cell has.

In addition to these cells the system consists of 3 additional counters called A, B and PC [2] . These counters contain pointers to certain cells in the system. Their exact function will be described below.

[1] this number is called "food" only for convenience, it has no connection to the notion of food we know. If we want to find analogies to this system in our universe, the whole system should be imagined as some big macromolecule, and each cell as a small molecular site with a molecule sitting in it. Food would then be the bonding energy of the molecule to the site.

[2] PC for Program Counter

2.1.2 The dynamics

The dynamics is very similar to a CPU acting on a computer memory. The following algorithm is used:

1. If the cell at the site of the PC has no food left then: The cell "dies" and a new cell "appears" at the same place, with a new, randomly chosen instruction and 7 units of food. The operation continues at step 2.
2. One unit of food is deducted from the amount which the cell at the PC has. Some action is taken according to the instruction in the cell at the PC.
3. The PC points to some new cell. For most types of instructions this is simply the next cell.

Steps 1–3 are repeated in a loop. each pass of steps 1–3 will be called a "generation".

As mentioned above there are 46 different instructions. These instructions can be classified into 10 main types with variants. The main types are:

give
take
copy
randB
setA
A+
A-
B+
B-
jmpA

The following actions are taken at step 3 of the algorithm, for the particular types of instructions. All except for jmpA cause the PC to be increased by one.

type of cell	*Action*
give	The cells pointed to by A, and the cells in its neighborhood are given a certain amount of food (2 units). There are 5 variants to give: give0, give1, give2 are all identical, the neighborhood of the cell is defined as 3 cells before to 3 cells after the cell. give3: The neighborhood is 4 cells before to 4 cells after the cell. give4: The neighborhood is 5 cells before to 5 cells after the cell.
take	Food from the cell pointed to by B and its neighborhood is taken and given to the cell pointed to by A and its neighborhood. As in 'give' there are 5 variants of 'take', which define the neighborhood. take0 – take2: 3 cell neighborhood. take3: 4 cell neighborhood. take4: 5 cell neighborhood.
copy	A neighborhood of cells around the cell pointed to by A is copied to the neighborhood of the cell pointed to by B. Each cell (at the destination) is given a certain amount of food (5 units). Again there are 5 variants of copy, defining the neighborhood to be copied. See give for the definition of the neighborhood.
rndB	Choose a new, random B. The variants of B define the distance from the PC in which the new B is chosen. rndB0 : choose B such that it is not farther then 500 cells away from the current PC. rndB1 : choose B such that it is not farther then 1000 cells away from the current PC. rndB2 — 1500 cells, rndB3 — 2000 and rndB4 3000 cells away from the current PC.

type of cell	*Action*
setA	Change A to point to the cell which is 5 cells behind the current PC. The 5 variants of setA (setA0–setA4) are all identical.
A+(A-)	move A by a certain distance forward or backward. variants: A+0: A moves to the next cell. A+1: A moves 2 cells forward. etc. A-0: A moves 2 cells backward. etc.
B+(B-)	These instructions are similar to A+(A-) with the difference that they move B.
jmpA	instead of increasing the PC by 1 after this instruction, the PC will be set to A. This type has only one variant: jmpA0.

The system is started with each cell containing a random instruction, and an initial amount of food (7 units).

3 Results

The simulation was run many times on a computer. The following account describes some characteristic results.

In several runs of the simulation some sequences of instructions which "take care" of themselves evolved. These instructions can be considered as a small computer program. The program enters an endless loop in which all its instructions are properly fed. In the next page the print-out of such a run can be seen. (this actually is the print-out of the first run of the simulation). Printed are cells 5544–6047 after $5 \cdot 10^6$ generations. For each cell the age, amount of food and instruction type is printed. (The age of a cell counts how many times the instruction in it was executed, since the cell was created).

Two programs, sets of instructions, are outlined. The first, which appeared at an earlier stage than the second, does the following: it copies itself to a random location, "steals" food from its child, chooses a new random location, copies itself there etc. . This program was quite successful — it appears "all over the place". The program probably dies when it accidently overwrites itself.

The second program outlined seems to be a mutation of the first. This program contains almost exactly the same instructions, instead of choosing a new, random location for its child, it gives itself more food. This program will not die, because it will never get overwritten: there is no chance that it will chose to overwrite itself, because it does not choose a new place to copy to. It will live forever, what might be called 'still life'. The result is an endless loop. Evolution has stopped. This kind of behavior was characteristic to most simulations of the system.

To find different kinds of behavior the laws of the system were changed. Instead of executing commands sequentially, several programs were executed in parallel: instead of having just one A, B and PC, there were now several sets of them. Thus in the case of two sets, action (steps 1–3 of the algorithm described) is taken first with the first set of A, B and PC, then with the second, then again with the first and so on. Each set of A, B and PC will be called a "processor". The case of one processor is the simulation just described . As the number of processors was increased a new behavior emerged. Of the 46 types of instructions 5 occupied more than 98the cells. The rest of the instructions were represented in very few cells. This shows that, with a high probability, almost no instruction had recently died of starvation: if many starved, they would have been changed randomly to one of the 46 types of instruction, and then every type would be represented.

The food is supplied in the following way: arbitrary sequences of cells are copied to random locations, there getting some amount of food (5 units). No cell dies because, at some stage, it will be overwritten by such a copy. This behavior also preserves the percentages of the various instructions. In this case no sequence of cells which can be called a "program" evolved. Only certain percentages and distribution of cell types did. Evolution could continue if the system evolved a more successful distribution of the instructions, or sequences of intructions, but this behavior was not observed.

World : MudRes0. starting from: 5544

584 4 rndB 1
584 4 jmpA 0
9 2 copy 2
9 3 setA 0
9 4 B + 0
1 4 B + 0
6 0 B + 2
6 5 B + 2
6 4 A + 3
6 4 B + 4
458 3 copy 3
455 3 take 3
454 4 rndB 1
454 3 jmpA 0
498 4 rndB 1
498 3 jmpA 0
9 3 B + 4
639 3 copy 3
635 3 take 3
634 3 rndB 1
634 2 jmpA 0
498 2 rndB 1
498 3 jmpA 0
9 3 B + 4
623 4 copy 3
619 4 take 3
618 4 rndB 1
618 6 jmpA 0
498 6 rndB 1
498 6 jmpA 0
9 6 B + 4
551 5 copy 3
547 4 take 3
546 4 rndB 1
546 4 jmpA 0
511 4 jmpA 0
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
629 5 copy 3
625 5 take 3
624 5 rndB 1
624 5 jmpA 0
8 5 setA 2
8 5 B + 3
8 6 rndB 1
3 6 copy 4
498 6 rndB 1
498 6 jmpA 0
9 6 B + 4
534 5 copy 3
530 4 take 3
529 4 rndB 1
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
583 2 copy 3
579 2 take 3
578 3 rndB 1
578 3 jmpA 0
498 3 jmpA 0
9 3 B + 4
562 3 copy 3
558 5 take 3
557 5 rndB 1
557 5 jmpA 0
7 26 B + 2
7 15 A + 3
7 5 A - 2
7 3 B + 4
7 3 take 1
7 4 A + 1

10 3 setA 3
10 3 give 1
10 3 B + 1
4 4 give 0
498 5 rndB 1
498 4 jmpA 0
9 4 B + 4
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
636 4 copy 3
632 4 take 3
631 4 rndB 1
631 4 jmpA 0
490 3 copy 3
486 3 take 3
485 3 rndB 1
485 2 jmpA 0
449 1 take 3
455 0 rndB 1
455 0 jmpA 0
203 3 copy 3
498 2 rndB 1
498 2 jmpA 0
9 3 B + 4
10 3 give 2
11 4 copy 4
4 3 give 0
4 3 give 2
239 4 take 3
238 3 rndB 1
6 2 B + 2
6 2 A + 3
6 2 B + 4
461 4 copy 3
9 5 B + 2
9 5 A + 3
9 6 B + 4
469 6 copy 3
465 6 take 3
498 6 rndB 1
498 6 jmpA 0
9 6 B + 4
522 6 copy 3
518 6 take 3
517 5 rndB 1
517 5 jmpA 0
463 5 rndB 1
463 5 jmpA 0
6 5 A + 3
6 4 B + 4
6 3 B + 2
6 4 A + 3
6 3 B + 4
309 2 copy 3
6 2 B + 2
6 1 A + 3
6 2 B + 4
464 3 copy 3
498 3 rndB 1
498 4 jmpA 0
9 4 B + 4
600 4 copy 3
596 5 take 3
595 5 rndB 1
595 5 jmpA 0
579 5 jmpA 0
1 6 take 4
2 6 B + 2
30 6 rndB 1
1 7 take 4
2 7 take 4

10 7 setA 3
10 7 B + 2
10 6 B + 2
38 6 rndB 1
10 6 B + 0
17 6 take 4
17 6 setA 3
17 5 B + 2
26 5 B + 0
14 5 B + 0
189 4 rndB 1
189 5 jmpA 0
1 7 B + 4
2 2 B + 4
6 63 B + 2
6 63 A + 3
6 63 B + 4
479 6 copy 3
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
631 5 copy 3
627 6 take 3
626 6 rndB 1
626 6 jmpA 0
498 6 rndB 1
498 5 jmpA 0
9 5 B + 4
620 5 copy 3
616 5 take 3
615 5 rndB 1
615 5 jmpA 0
3 5 A + 0
3 6 B + 2
498 6 rndB 1
498 6 jmpA 0
9 6 B + 4
532 6 copy 3
528 5 take 3
527 3 rndB 1
527 3 jmpA 0
22 7 setA 3
498 3 rndB 1
498 3 jmpA 0
9 3 B + 4
557 3 copy 3
553 5 take 3
552 5 rndB 1
552 5 jmpA 0
524 5 copy 3
520 5 take 3
519 5 rndB 1
519 5 jmpA 0
4 5 setA 4
4 5 setA 4
10 4 B + 2
10 3 A + 3
8 3 B + 2
498 3 rndB 1
498 2 jmpA 0
9 2 B + 4
597 2 copy 3
593 3 take 3
592 4 rndB 1
498 4 rndB 1
498 4 jmpA 0
9 5 B + 4
602 4 copy 3
598 3 take 3

597 3 rndB 1
597 3 jmpA 0
8 6 rndB 1
8 0 copy 4
8 3 rndB 3
2 8 take 4
2 10 B + 0
3 8 copy 1
3 7 take 3
3 7 A + 0
498 5 rndB 1
498 5 jmpA 0
9 5 B + 4
650 5 copy 3
646 4 take 3
645 5 rndB 1
645 5 jmpA 0
8 26 B + 2
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
514 5 copy 3
510 5 take 3
509 4 rndB 1
542 6 jmpA 0
498 6 rndB 1
498 6 jmpA 0
9 6 B + 4
11 6 give 2
12 6 copy 4
4 6 give 0
4 6 give 2
4 5 give 0
9 4 B + 2
498 3 rndB 1
498 3 jmpA 0
9 2 B + 4
542 4 jmpA 0
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
3 5 give 2
4 4 copy 4
12 5 B + 4
501 5 rndB 1
501 5 jmpA 0
457 5 jmpA 0
9 5 A + 3
9 5 B + 4
455 6 copy 3
451 6 take 3
450 6 rndB 1
450 6 jmpA 0
1 5 A - 2
1 5 setA 4
7 3 B + 4
7 4 take 0
7 4 B + 1
542 3 jmpA 0
498 3 rndB 1
498 4 jmpA 0
9 4 B + 4
505 4 rndB 1
1 4 copy 4
9 4 B + 4
498 5 rndB 1
498 5 jmpA 0
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
547 4 copy 3
543 3 take 3

542 4 rndB 1
542 4 jmpA 0
498 177 rndB 1
498 179 jmpA 0
9 285 B + 4
21 208 give 2
22 71 copy 4
15 16 give 0
15 14 give 2
7 0 copy 2
8 0 B + 4
8 0 copy 1
1 6 give 1
1 5 rndB 3
1 6 give 3
1 6 take 1
6 4 rndB 2
1 5 rndB 3
9 5 B + 2
7 5 B + 2
7 5 A + 3
7 5 B + 4
449 5 copy 3
446 6 take 3
8 6 copy 1
8 5 take 4
2 5 A - 0
4 5 setA 4
4 5 setA 4
10 5 B + 2
10 6 A + 3
9 6 take 4
8 6 A + 1
11 5 B + 2
11 526 A + 3
11 1659388 B + 4
484 1659385 copy 3
1659862 3317722 take 3
1658860 8 give 3
1659860 4 jmpA 0
9 524 B + 4
466 4 copy 3
11 3 A + 3
11 3 B + 4
484 4 copy 3
479 3 take 3
478 2 rndB 1
478 3 jmpA 0
9 2 B + 4
9 3 B + 4
508 3 copy 3
504 2 take 3
503 3 rndB 1
503 4 jmpA 0
397 4 copy 3
393 5 take 3
392 5 rndB 1
139 0 take 3
9 5 B + 2
9 5 A + 3
9 5 B + 4
480 5 copy 3
476 5 take 3
475 5 rndB 1
475 6 jmpA 0
7 6 copy 1
7 6 take 4
1 4 A - 0
6 4 setA 4
6 4 setA 4
12 4 B + 2
498 4 rndB 1

498 3 jmpA 0
9 3 B + 4
520 5 copy 3
516 5 take 3
515 4 rndB 1
515 4 jmpA 0
1 5 take 4
1 6 setA 3
6 5 B + 2
498 4 rndB 1
498 4 jmpA 0
9 5 B + 4
621 5 copy 3
617 5 take 3
616 5 rndB 1
616 4 jmpA 0
9 5 B + 4
498 5 rndB 1
498 5 jmpA 0
9 5 B + 4
641 5 copy 3
637 5 take 3
636 5 rndB 1
636 4 jmpA 0
6 4 A + 3
2 5 take 1
1 5 B + 2
7 5 A - 2
7 4 setA 2
8 5 B + 3
8 6 rndB 1
8 6 copy 4
7 6 rndB 3
498 5 rndB 1
498 5 jmpA 0
9 6 B + 4
550 5 copy 3
498 4 rndB 1
498 4 jmpA 0
9 3 B + 4
627 3 copy 3
498 3 rndB 1
498 2 jmpA 0
9 3 B + 4
645 3 copy 3
641 2 take 3
640 2 rndB 1
640 2 jmpA 0
498 2 rndB 1
498 2 jmpA 0
9 2 B + 4
619 2 copy 3
615 2 take 3
614 3 rndB 1
614 3 jmpA 0
6 3 B + 2
6 4 A + 3
6 3 B + 4
392 5 rndB 1
7 4 B + 1
9 4 B + 2
9 5 B + 2
9 5 A + 3
9 5 B + 4
9 4 B + 2
9 3 A + 3
9 3 B + 4
485 3 copy 3
481 3 take 3
480 2 rndB 1
480 3 jmpA 0
472 4 jmpA 0
6 5 A + 3
6 4 B + 4
449 3 copy 3
446 3 take 3

445 3 rndB 1
445 2 jmpA 0
498 1 rndB 1
498 1 jmpA 0
9 2 B + 4
604 2 copy 3
600 2 take 3
599 2 rndB 1
599 3 jmpA 0
561 3 copy 3
557 3 take 3
556 3 rndB 1
556 4 jmpA 0
301 3 rndB 1
498 4 rndB 1
498 4 jmpA 0
9 4 B + 4
652 3 copy 3
648 3 take 3
647 3 rndB 1
647 4 jmpA 0
469 1 take 3
9 4 B + 2
9 4 A + 3
9 5 B + 4
484 5 copy 3
480 5 take 3
479 5 rndB 1
479 3 jmpA 0
498 3 rndB 1
498 4 jmpA 0
9 4 B + 4
534 5 jmpA 0
12 4 B + 4
4 4 take 1
3 5 give 1
3 5 B + 1
6 4 B + 4
362 4 copy 3
359 5 take 3
358 6 rndB 1
358 6 jmpA 0
6 6 B + 2
6 6 A + 3
6 3 B + 4
273 2 copy 3
6 2 B + 2
6 2 A + 3
6 2 B + 4
355 2 copy 3
352 2 take 3
351 5 rndB 1
351 6 jmpA 0
6 6 B + 2
6 6 A + 3
6 6 B + 4
454 6 copy 3
451 6 take 3
450 6 rndB 1
450 5 jmpA 0
6 5 B + 4
415 5 copy 3
412 3 take 3
411 3 rndB 1
411 3 jmpA 0
1 4 take 4
2 4 B + 2
30 3 rndB 1
2 3 B + 0

name	n	average age	average food	max age	max food
give0	0				
give1	0				
give2	0				
give3	0				
give4	1	5	7	5	7
take0	0				
take1	0				
take2	4	34	6	35	7
take3	0				
take4	0				
copy0	0				
copy1	0				
copy2	0				
copy3	0				
copy4	2679	377	6	437	7
rndB0	2143	301	6	361	7
rndB1	0				
rndB2	0				
rndB3	0				
rndB4	11	12	5	16	7
setA0	896	416	6	521	7
setA1	6	5	6	7	7
setA2	310	360	6	471	7
setA3	0				
setA4	0				
A+0	2	1	7	1	7
A+1	0				
A+2	0				
A+3	1943	419	6	561	7
A+4	0				
A-0	0				
A-1	0				
A-2	2	2	6	2	6
A-3	0				
A-4	0				
B+0	0				
B+1	0				
B+2	0				
B+3	0				
B+4	0				
B-0	0				
B-1	0				
B-2	0				
B-3	0				
B-4	0				
jmpA0	0				

Table 1: State of at the end of a run with several processors

4 discussion

When one examines the results of the simulation, the first question one should ask is whether Evolution actually occured. Can the process which resulted be called Evolution? In the first type of simulation (with one processor) almost every run ended with a program in which an endless loop feeds itself. The chance of such a program appearing at the initial state (the begining of the simulation) is much smaller than 100%. The chance of any particular sequence of 4 instructions to appearing the initially is 0.2%, the chance for a 5 instruction sequence is $< 4 \cdot 10^{-3}\%$ and self feeding programs are usually longer than that.

Thus the system houses some process which develops most settings to endless loops. For a system which is not conserving in phase-space [3] (and at first sight our system is not conserving [4]) this is not very surprising. Such a process exists in any system with a sink or any other convergence. In systems which are conserving in phase space this is not possible. In conserving systems there can be no convergence towards some selected volume in phase-space. Nevertheless, in these systems there can be convergence in some of the degrees of freedom, and divergence in the others. The laws of our universe are conserving in phase-space, and Evolution is a process which causes convergence in some of the degrees of freedom, and divergence in the others. (Consider the solution to the question 'Why doesn'st evolution contradict the law of entropy?') In the third result, convergence is even more apparent. In the end-state only 5 out of 46 instructions were represented. The chance of this to occuring in the initial state is astronomically small. This 3rd result seems more like some simple convergence or crystallization than the first results. Convergence probably is one indicator of Evolution, yet some methods must be found to distinguish between 'simple' convergence and Evolution.

It is apparent that there was some feature in the system which led to the evolution of complexity. If we consider this to be a seed

[3] for discrete systems conserving in phase-space means being reversible

[4] but the system of the whole universe, with our computer and the simulation in it, is conserving

of Evolution, we see that some of the ingredients of the Evolution we know from biology are missing: the process of Evolution stopped after a short time.

Biological Evolution seems to last forever, or for a very long time. Through the process of biological Evolution creatures of considerably more complexity than any of their surrounding evolved. It is very important to find conditions (or build simulations) which are necessary for such long lasting Evolution. There is one considerable problem which could prevent a computer simulation from housing such Evolution. Our system, or any computer-simulation, is very small when compared to a chemical system. Maybe the amount of 'information which can be gained' from our system limits the Evolution to a very short period. A small environment on earth contains an enormous amount of molecules. In a cup of water, for example there are about 10^24 molecules. These can contain an amount of information we cannot hope to have in any computer simulation. It has to be determined whether Evolution exists only in such big systems.

Self replication is considered essential for Evolution. In our system, in not all cases did evolution progress through self-replicating forms, and when it did, self-replicating forms were not always the end-product (the fittest). Only one of the 3 types of behavior described can be considered as classical 'self replication'. This is the program which copies itself to various locations. The second behavior, the endless loop, could be considered as self replication when one looks at the time dimension. The behavior which evolved when there were many processors has no self-replication, it is some kind of growth.

Another subject to be considered is the question of fitness. Which were the fittest creatures? One can see that in the first system (with one processor) the competition was mainly for processor time. The fittest programs were those which managed to conquer most of the CPU time. In the second case, with many processors, it seems that processor time is abundant, and the competition is for space, or cells. The program which can conquer most cells wins. In section 2 I described the principles for building the simulation. The third stated that fitness should not

be externally defined, but should be developed "naturally" by the system. The two cases of fitness which evolved show us that the question of the type of fitness which evolves in any particular dynamical system is interesting. In population genetics "fitness" has a well defined meaning, but in a general evolution model, different types of competitive ability may be encountered. Which of these could become the fitness of the system? This is probably one of the first questions which should be considered.

One thing which was learned from this model is that one should not think of Evolution as one unit. It seems that the term 'Evolution' includes many subprocesses which may contribute to what we call Evolution.

1. Self replication. Self replication makes the gathering of information easier. Probably most Evolutions develop at some stage self replication and some information carrier. Still there might exist models of Evolution in which there is no self replication for some period. This could be achieved by using some other forms of cycles (see Eigen [**?**]) or non-cyclic growth/change (see Dyson [**?**])
2. Many possibilities. At various stages of Evolution some "solution" to a "problem" has to be found. Often each solution has a very low probability to appear, but many different solutions exist and thus one path may be chosen. In our system this could be seen when in the first type of results there were many types of self-replicating/self-feeding programs which could appear.
3. Change of environment - internal. The process of Evolution changes the environment. The raw environment is not suited for complex creatures, and Evolution creates a new, proper environment.
4. Change of environment - external. in biological Evolution change of the environment seems to have contributed much to Evolution. It probably causes the system to exit from "local maxima" of fitness. (Dinosaurs - mammals) In the system described there was no external change of environ-

ment, because of rule 3. Future simulations will probably deal with this.

5. Shaping of the system. In the process of Evolution new laws governing the system are created. The system is, of course, still governed by its dynamics, but the exact nature of this dynamics is not important to Evolution. Thus, after the formation of self-reproducing units and aquisition of some type of fitness, Evolution could be governed by laws like 'survival of the fittest', laws of game-theory or laws of population genetics. All this will occur without much influence by the internal dynamics. When these laws take over, Evolution can play on its home ground.

Most models of the origin of life try to explain this origin from a chemical view point, and when a more abstract viewpoint is chosen it is taken with a chemical background. I belive that it is also important to consider models of Evolution which do not have a chemical basis. The model presented here is a preliminary attempt to construct such a model.

References

[1] M.Eigen *Naturwissenshaften* 58 465 (1971).

[2] M Eigen and P. Schuster, *Naturwissenschaften* 64 541 (1977); 65 7 (1978); 65 341 (1978).

[3] F.J.Dyson *J.Mol.Evol* 18 344 (1982); *Origins of Life* (Cambridge: Cambridge University Press, 1985).

[4] J.F. Crow and M.J. Kimura *An Intr. to Population Genetics* (New York: Harper&Row, 1970).