Published in: Advances in Artificial Life, Proc. 3rd European Conf. on Artificial Life, Grenada, Spain, June 4-6, 1995, Lecture Notes in Artificial Intelligence 929, Springer, Berlin, 1995.

A New Self-Reproducing Cellular Automaton Capable of Construction and Computation

Gianluca Tempesti

Logic Systems Laboratory, Swiss Federal Institute of Technology
INN-Ecublens, CH-1015 Lausanne, Switzerland

Abstract

We present a new self-reproducing cellular automaton capable of construction and computation beyond self-reproduction. Our automaton makes use of some of the concepts developed by Langton for his self-reproducing automaton, but provides the added advantage of being able to perform independent constructional and computational tasks alongside self-reproduction. Our automaton is capable, like Langton's automaton and with comparable complexity, of simple self-replication, but it also provides (at the cost, naturally, of increased complexity) the option of attaching to the automaton an executable program which will be duplicated and executed in each of the copies of the automaton. After describing in some detail the self-reproduction mechanism of our automaton, we provide a non-trivial example of its constructional capabilities.

1 Introduction

The history of self-reproducing cellular automata basically begins with John von Neumann's research in the field of complex self-reproducing machines. Advised by the mathematician Stan Ulam, he applied his concepts in the framework of a "cellular space", a two-dimensional grid of identical elements where each element (cell) is a finite state automaton whose next state is a function of its present state and of the present state of its 4 neighboring cells.

Within this framework, von Neumann was able to conceive a self-reproducing automaton endowed with the properties of both computational and constructional universality [1]. Unfortunately, the automaton was of such complexity that, further simplifications notwithstanding, even today's state-of-the-art computers lack the power to simulate it in its entirety.

The next significant event in the history of self-reproducing automata was the development of the automaton commonly referred to as "Langton's loop" [2]. By dropping the requirements of computational and constructional universality, Langton created an automaton capable of non-trivial self-replication, that is an automaton where the replication is actively directed by the automaton itself, rather than being a mere consequence of the transition rules.

The automaton we introduce seeks to go beyond Langton's loop, which is capable exclusively of duplicating itself, by adding computational and constructional capabilities to self-reproduction. In fact, while our automaton is based on the utilization of a "loop" similar to that of Langton's automaton, we have modified the self-reproducing mechanism so that it requires only a fraction of the data circulating in the loop to perform its task, thus making the remaining data available for other purposes.

In the next chapter, we will present an overview of the cellular automata mentioned above, and compare them with our own automaton. We will then describe in detail the operation of our automaton, and provide an example of its constructional capabilities.

2 Self-reproducing cellular automata

2.1 Von Neumann's automaton

Von Neumann's self-replicating cellular automaton was a result of the mathematician's interest in complex machines and their behavior [1]. His research led to the conclusion that the following characteristics should be present in a self-reproducing machine:

- Computational universality, that is the ability to operate as a universal Turing machine, and thus to execute any computational task.
- Constructional universality, that is the ability to construct any kind of configuration in the cellular space starting from a given description; self-reproduction is then a particular case of universal construction.

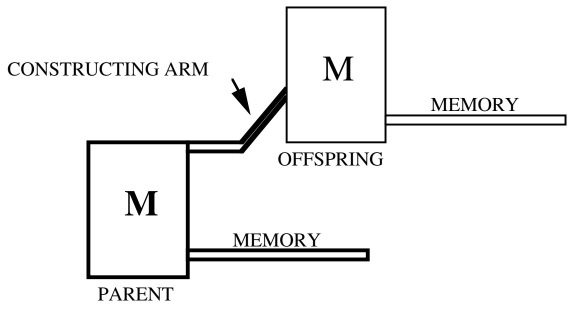


Fig. 1: Von Neumann's self-reproducing automaton

To implement these properties in a cellular automaton, von Neumann set out to design a universal constructor, i.e. an automaton capable of constructing, through the use of a "constructing arm", any configuration whose description can be stored on its input tape (Fig. 1). This universal constructor, therefore, is able, given its own description, to construct a copy of itself, thus achieving self-reproduction.

The automaton developed by von Neumann used tens of thousands of 29-state cells and a 5-cell neighborhood (the cell itself plus its four cardinal neighbors). Codd [3] and others managed to reduce the complexity of von Neumann's machine, but the automaton retains a level of complexity too high for simulation. In fact, while parts of the machine have been successfully simulated, the task of simulating the whole automaton remains virtually impossible given current technology.

2.2 Langton's loop

Langton's automaton [2] is based on one of the components of Codd's universal constructor, namely the "periodic emitter" [3]. The automaton (Fig. 2) is essentially a square loop, with internal and external sheaths, where the data necessary for the construction of a duplicate loop circulate counterclockwise. Duplication is achieved by extending a constructing arm which will be forced to turn 90 degrees to the left at regular intervals corresponding to the size of one side of the loop. After three such turns the arm will have folded upon itself. When the new loop is closed the constructing arm will retract and the new loop will be active, that is will be able to reproduce itself as the original loop did. The original loop will then repeat the process by creating a second copy of itself in another direction, and finally "die" by losing the information within the loop. Given sufficient time, the automaton will replicate itself to fill the available space.

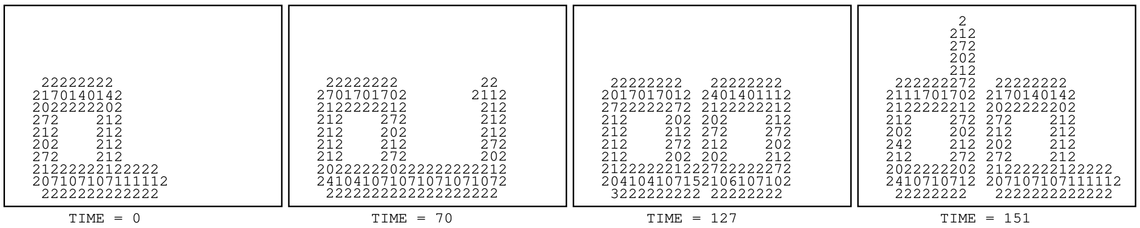


Fig. 2: Langton's Loop

Langton's loop uses 8 states for each of the 86 non-quiescent cells making up its initial configuration, a 5-cell neighborhood, and a few hundred transition rules (the exact number depends on whether default rules are used and whether rotated rules are included in the count). Further simplifications to the automaton were introduced by Byl [5], who eliminated the internal sheath and reduced the number of states per cell, the number of transition rules, and the number of non-quiescent cells in the initial configuration. Reggia et al. [6] managed to remove also the external sheath. Given their low complexity, at least relative to von Neumann's automaton, all of the mentioned automata have been thoroughly simulated.

2.3 The new automaton

Our automaton uses some of the concepts found in Langton's loop. In particular, we retain the concept of loop, which Langton himself derived from Codd's periodic emitter, to store the data dynamically. However, there are some substantial differences between our loop and Langton's automaton (Fig. 3):

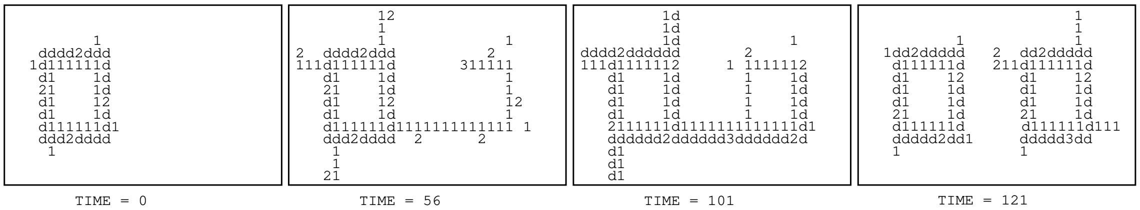


Fig. 3: Our Loop

- We use a 9-cell neighborhood (the cell itself plus its 8 neighbors).
- As in Byl's version of Langton's loop, we use only one sheath, but contrary to Byl, we retain the internal sheath and eliminate the external one. This allows us to let the data in the loop circulate without the need for leading or trailing states (the 0s in Langton's loop). In addition to the internal sheath, we have four "gate cells" (in the same state as the sheath) outside the loop at the four corners of the automaton. These cells are initially in the "open" position, and will shift to the "closed" position once the copy is accomplished.
- We extend four constructing arms in the four cardinal directions at the same time, and thus create four copies of the original automaton in the four directions in parallel. When the arm meets an automaton already in place where the copy should be (which happens for all but the original automaton), it simply retracts and puts the corresponding gate cell in the closed position.
- Rather than being directed to advance, our constructing arm advances by default. As a consequence, it is necessary only to direct it to turn at the appropriate moment. This is done by sending periodic "messengers" to the tip of the constructing arm, which advanced at a slower pace with respect to the messengers.
- The arm does not immediately construct the entire loop. Rather, it constructs a sheath of the same size as the original. Once the sheath is ready, the data circulating in the loop is duplicated and the copy is sent along the constructing arm to wrap around the new sheath. When the new loop is completed, the constructing arm retracts and shifts the corresponding gate cell to the closed position.
- As a consequence of the above, rather than using all of the data in the loop to direct the constructing arm, we use only four of the cells circulating in the loop to generate the messengers. Since the only operation performed on the remaining data cells is duplication, they do not have to be in any particular state. In particular, they can be used as a "program", i.e., a set of states with their own transition rules which will then be applied alongside the self-reproduction to execute some function.
- Unlike Langton's loop, our loop does not "die" once duplication is achieved, as the circulating data remains untouched by the self-reproduction process. Therefore, any program stored in the loop will be able to continue to execute. Also, it is possible to force the loop to try and duplicate again in any of the four directions simply by shifting the corresponding cell back to the open position.
- When the duplicated loops arrive next to the border of the array, the constructing arm detects the border and retracts without attempting to duplicate the data. Thus, our automaton, unlike Langton's, does not crash when the duplication process reaches the edge of the cellular space.
- Because the reproduction process occurs in the four directions at the same time, the growth of the colony follows a symmetric pattern (Fig. 4), unlike the spiraling pattern of Langton's automaton.

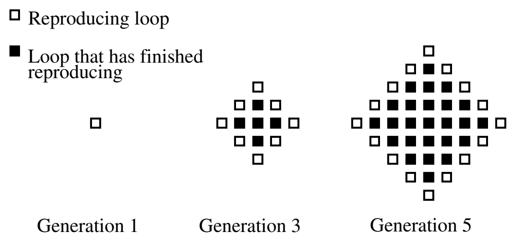


Fig. 4: Growth pattern

As should be obvious from the above, while our loop owes to von Neumann the concept of constructing arm and to Langton (and/or Codd) the basic loop structure, it is in fact a very different automaton, endowed with some of the properties of both.

As far as the complexity of the automaton is concerned, its estimation is more difficult than for Langton's loop, as it depends on the data circulating in the loop. The number of non-quiescent cells making up the initial configuration depends directly on the size of the circulating program. The more complex (i.e. the longer) the program, the larger the automaton. It should be noted, however, that the complexity of the self-reproduction process does not depend on the size of the loop. The number of states also depends on the complexity of the program. To the 5 "basic states" used for self-reproduction (see description below) must be added the "data states" (at least one) used in the program, which must be disjoint from the basic states. The number of transition rules is obviously a function of the number of data states: in the basic configuration (i.e., one data state), the automaton needs 692 rules (173 rules rotated in the four directions). By default, all cells remain in the same state.

The complexity of the basic configuration is therefore in the same order as that of Langton's and Byl's loops, with the proviso that it is likely to increase drastically if the data in the loop is used for some purpose. In fact, the number of rules in the automaton we have described grows as D^4, where D is the number of data states. A different version of the automaton limits the growth to D^3 (at the expense of some versatility), but the increase remains substantial.

In the next chapter we will describe in some detail the operation of the automaton in a small, basic configuration, and illustrate an example of a loop where a program has been included in the loop to demonstrate the construction capabilities of our automaton.

3 Description of the automaton

3.1 Cellular space and initial configuration

As for von Neumann's and Langton's automata, the ideal cellular space for our automaton is an infinite two-dimensional grid. Since we realize that a practical implementation of such a cellular space might prove difficult, we added some transition rules to handle the collision between the constructing arm and the border of the array. On meeting the border, the arm will retract without attempting to make a copy of the parent loop.

The cells of the array require five basic states and at least one data state (see Fig. 4 at time 0). State 0 is the "quiescent state" and is represented by a blank space in the figures. State 1 is the "sheath state", that is the state of the cells making up the sheath and the four gates. State 2 is the "activation state". The four cells in the loop directing the reproduction are in state 2, as are the messengers which will be used to command the constructing arm and the tip of the constructing arm itself for the first phase of construction, after which the tip of the arm will pass to state 3, the "construction state". State 3 will construct the sheath that will receive the copy of the loop, signal the parent loop that the sheath is ready, and lead the duplicated data to the new loop. State 4, the "destruction state", will destroy the constructing arm once the copy is ready. In addition to these states, we have labeled 'd' the data state, with the understanding that this one symbol might in fact represent any set of states not including states 0 to 4.

The initial configuration is in the form of a square loop wrapped around a sheath. The size of the loop is a variable that for our example have set to 8x8. The loop is a sequence of data states in which four cells in the activation state are placed at a distance from each other equal to the side of the loop. Near the four corners of the loop we have placed four cells in the sheath state. These are the gate cells, and the position they occupy signifies that the gates are open (that is, that the automaton should attempt to duplicate itself in all four directions).

3.2 Operation

Once the cellular space starts operating, the data starts turning around the loop. Nothing happens until the first 2 reaches a corner, where it finds the gate open. Since the gate is open, the

2 splits into two identical cells. One cell continues turning around the loop, while the second starts extending the arm (Fig. 5a). The arm advances by one cell each two time periods. Once the arm has started extending, each 2 that arrives to a corner will again split and one of the copies will start running along the arm, advancing by one cell per cycle (Fig. 5b). Since the arm is extending at half the speed of these messengers and the messengers are spaced 8 cells apart (the length of one side of the loop), the messengers will reach the tip of the arm at regular intervals corresponding to the length of one side of the loop.

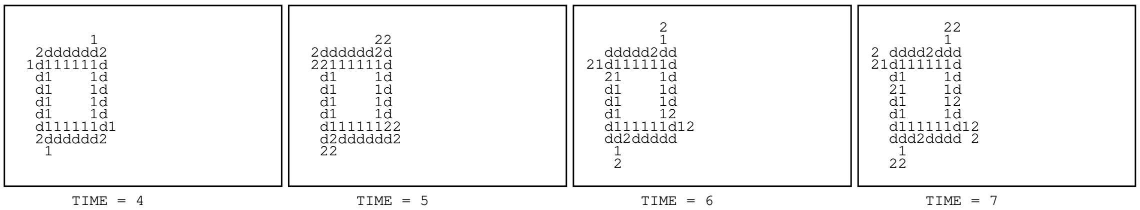


Fig. 5a: The constructing arm starts extending

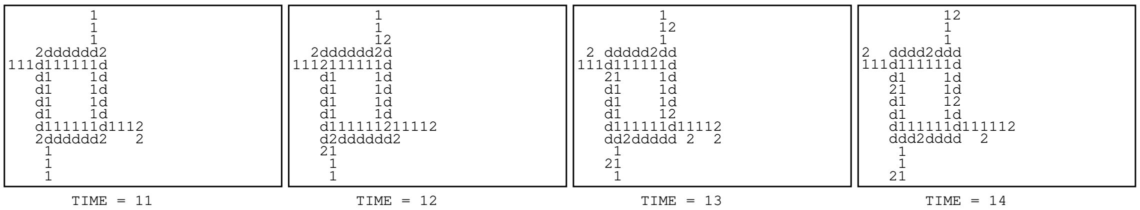


Fig. 5b: The first messenger leaves the loop

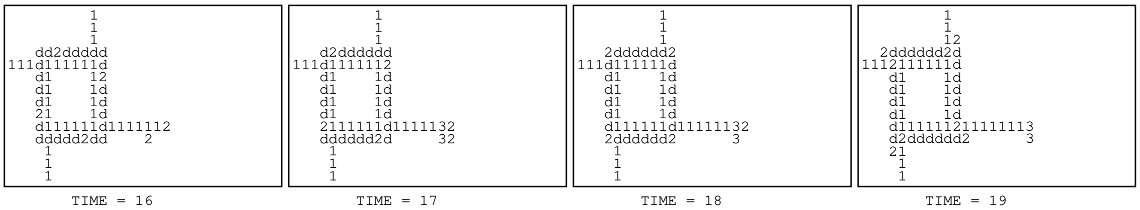


Fig. 5c: The first messenger reaches the tip of the constructing arm

When the first messenger reaches the tip of the arm, the tip, which was until then in state 2, passes to state 3 and continues to advance at the same speed (Fig. 5c). This transformation tells the arm that it has reached the location of the offspring loop and to start constructing the new sheath.

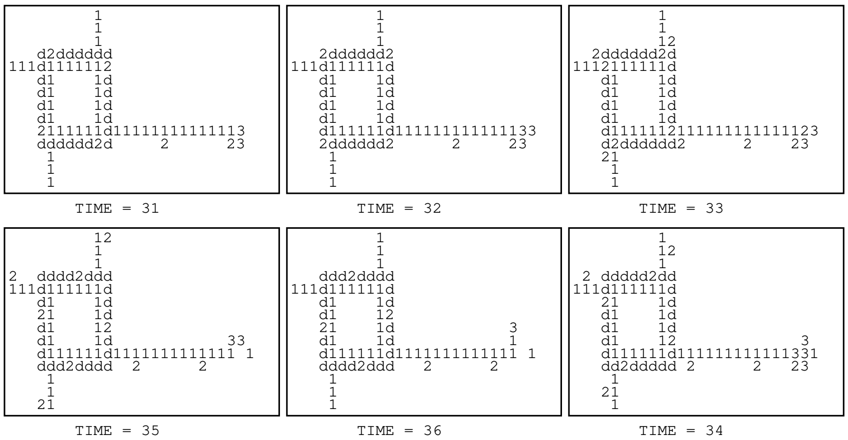


Fig. 5d: The second messenger forces the arm to turn to the left

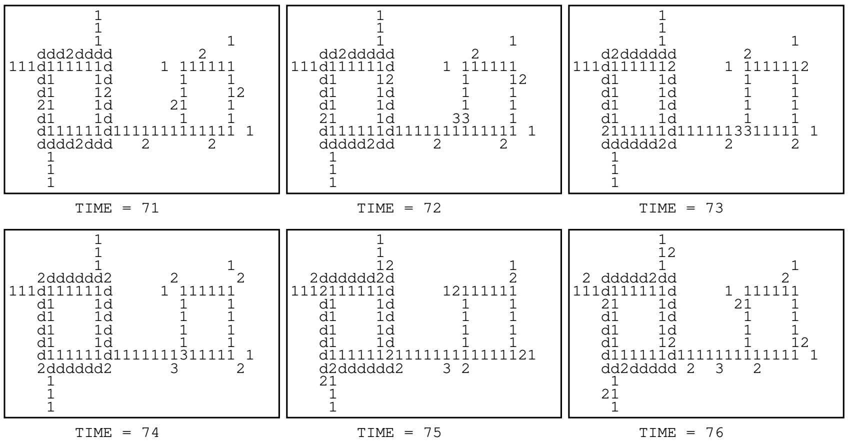


Fig. 5e: The arm closes the new loop

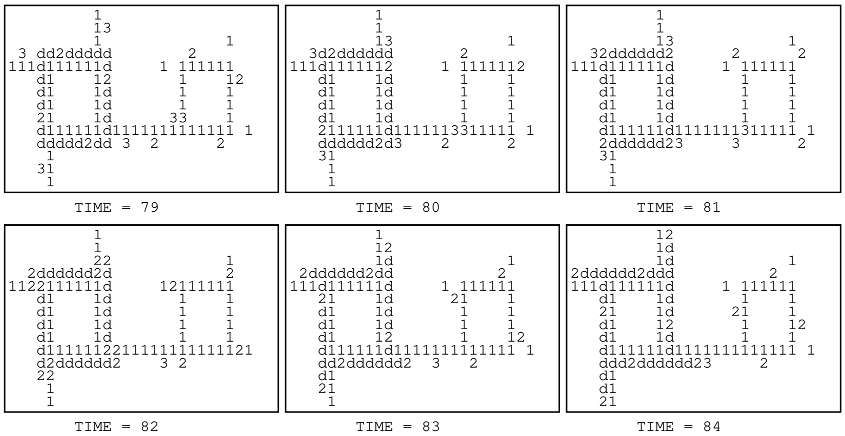


Fig. 5f: The return signal starts the copy of the data

The next two messengers will force the tip of the arm to turn left (Fig. 5d), while the fourth will reach the tip as the arm is closing upon itself (Fig. 5e). It causes the sheath to close and then runs back along the arm to signal to the original loop that the new sheath is ready.

Once the return signal arrives at the corner of the original loop, it waits for the next 2 to arrive (Fig. 5f). When the 2 sees the 3 waiting by the gate, again it splits, one copy staying around the loop, the other running along the arm. This time, however, rather than running along the arm in isolation as a messenger, it carries behind him a copy of the data in the loop. In the copy, the activation cells are temporarily switched off (set to state 3) until the new sheath is reached, where they will again become 2s and start their function.

Always followed by the data, it runs around the sheath until it has reached the junction where the arm folded upon itself (Fig. 5g). On reaching that spot, it closes the loop and sends a destruction signal (the 4) back along the arm. The signal will destroy the arm until it reaches the corner of the original loop, where it closes the gate (Fig. 5h).

Meanwhile, the new loop is already staring to reproduce itself in three of the four directions. One direction (down in the figures) is not necessary since another of the new loops will always get there first, and therefore its corresponding gate will be set to the closed position.

After 121 time periods the gates of the original automaton will be closed and it will enter an inactive state, with the understanding that it will be ready to reproduce itself again should the gates be opened.

```
         1d            2
         1d            1
         1d            1
dddddd2dddd      ddd2dddd
111d111111d     1d111111d
   d1    1d      d1    1d
   d1    12      d1    12
   21    1d      21    1d
   d1    1d       1    1d
   d111111d1111111111111d111
   dddd2dddddd3dddddd3ddd
   d1
   d1
   d1
```

TIME = 113

```
         1d            22
         1d            1
         1d            1
ddddd2ddddd      dd2ddddd
111d111111d     1d111111d
   d1    12      d1    12
   d1    1d      d1    1d
   d1    1d      d1    1d
   21    1d      33    1d
   d111111d1111111111111d111
   ddddd2dddddd3dddddd3dd
   d1
   d1
   d1
```

TIME = 114

```
         1d            1
         1d            1
         1d            1
dddd2dddddd      d2dddddd
111d1111112     1d1111112
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d     4d1    1d
   2111111d1111111111111d111
   dddddd2dddddd3dddddd2d
   d1
   d1
   d1
```

TIME = 115

```
         1d            1
         1d            1
         1d            1
ddd2dddddd2      2dddddd2
111d111111d     1d111111d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d111111d111114d111111d111
   2dddddd2dddddd3dddddd2
   d1
   d1
   d1
```

TIME = 116

```
         1d            1
         1d            1
         13            12
dd3dddddd2d     2dddddd2d
1112111111d     22111111d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d111111211114 d1111112111
   d2dddddd3dddd d3dddddd2
   31
   d1
   d1
```

TIME = 117

```
         1d            1
         13            12
         1d            1
d3dddddd2dd      ddddd2dd
111d111111d    21d111111d
   21    1d      21    1d
   d1    1d      d1    1d
   d1    1d      d1    1d
   d1    12      d1    12
   d111111d1114  d111111d111
   dd2dddddd3dd  dd3ddddd 2
   d1            1
   31
   d1
```

TIME = 118

Fig. 5g: The data wraps around the new loop and the arm is destroyed

```
                      1
        4d            1
        1d            1
ddddd2dddd      ddd2dddd
41d111111d   211d111111d
  d1    1d      d1    1d
  d1    12      d1    12
  21    1d      21    1d
  d1    1d      d1    1d
  d111111d14    d111111d111
  dddd2ddddd    dddd3ddd
  d1            1
  d4
```

TIME = 120

```
                      1
                      1
         1            1
 1dd2ddddd   2  dd2ddddd
  d111111d   211d111111d
  d1    12      d1    12
  d1    1d      d1    1d
  d1    1d      d1    1d
  21    1d      21    1d
  d111111d      d111111d111
  ddddd2dd1     ddddd3dd
  1             1
```

TIME = 121

Fig. 5h: The original cell becomes inactive

3.3 Example

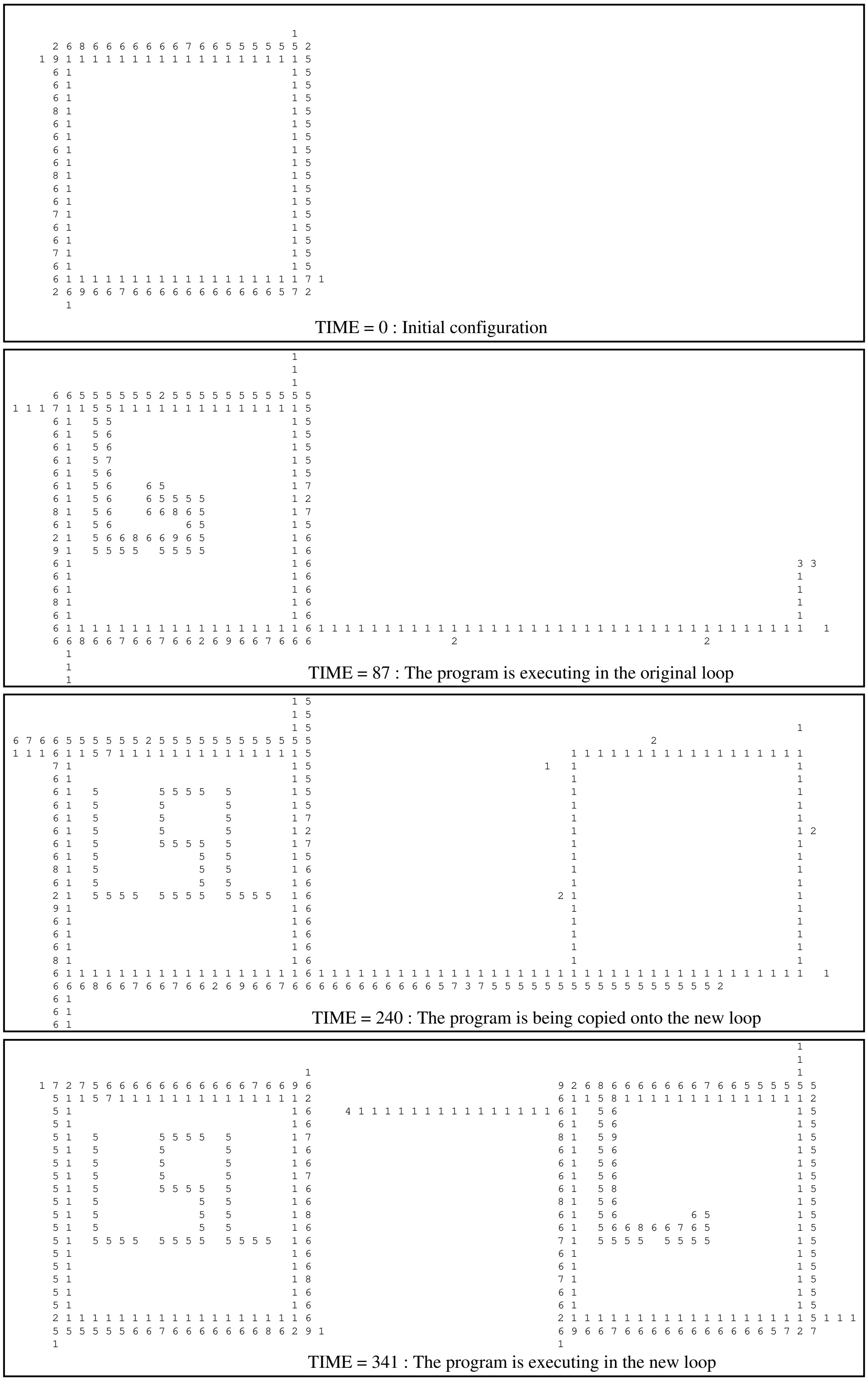
In fig. 6, we illustrate an example of how the data states can be used to carry out operations alongside self-reproduction. The operation in question is the construction of three letters, LSL (the acronym of Logic Systems Laboratory), in the empty space inside the loop. Obviously this is not a very useful operation from a practical point of view, but it is a non-trivial case of construction that should demonstrate some of the capabilities of the automaton.

For this example, we have used 5 data states, which have brought the number of transition rules to 35202. Of these, 326 are new rules which control the behavior of the program, and do not concern self-reproduction. The loop size is 20x20, and a full reproduction of a loop requires 321 time periods.

The operation of the program is fairly straightforward. When a certain "initiation sequence" within the loop arrives to the top left corner of the loop, a "door" is opened in the internal sheath. The rest of the program, as it passes by the door in its rotation around the loop, is duplicated and one of the copies enters the interior of the loop, where it is treated as a sequence of instructions which direct the construction of the three letters. The construction mechanism is somewhat similar to the method Langton used in his own loop, with single-cells instructions such as "turn left", "advance", etc. The construction ends when a "termination sequence" arrives at the door. At that stage, the door is closed and a flag is set in the sheath to warn that the program has already executed.

During the process of reproduction, the program is simply copied (as opposed to interpreted as in the interior of the cell) and arrives intact in the new loop, where it will execute again exactly as it did in the parent loop.

This is a simple demonstration of one way in which the data in the loop could be used as an executable program. Many other methods can be envisaged, and we are currently working on the development of other, hopefully more useful, programs. Our ultimate goal would be to be able to construct, using a program stored in one of our loops, a universal Turing machine. Some of the features of such machines render this a difficult task which will probably require a modification of the basic mechanisms of our automaton, but we are confident that such a construction is indeed feasible.



TIME = 0 : Initial configuration

TIME = 87 : The program is executing in the original loop

TIME = 240 : The program is being copied onto the new loop

TIME = 341 : The program is executing in the new loop

Fig. 6: An example of the capabilities of our automaton

4 Conclusion

We have described a new self-reproducing cellular automaton which provides some of the features of both von Neumann's and Langton's automata. Our automaton is capable, like Langton's loop and with comparable complexity, of simple self-replication, but it also provides (at the cost, naturally, of increased complexity) the option of attaching to the automaton an executable program which will be duplicated and executed in each of the copies of the loop.

The example we illustrated, while not trivial in its complexity, is far from being a full demonstration of the capabilities of the automaton.

In particular, we have shown that our loop is capable of *some* construction beyond simple self-reproduction, and thus, trivially, of *some* computation. However, we have not yet fully investigated the limits of the automaton as a constructor. That is, we have not yet determined whether our machine is capable of constructional universality. Should that be the case, computational universality should follow trivially (since the automaton would be able to construct a universal Turing machine), and the properties outlined by von Neumann for self-reproducing machines would be met.

References

1. J. von Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, ed., University of Illinois Press, Urbana, 1966.
2. C. G. Langton, "Self-Reproduction in Cellular Automata", *Physica 10D*, pp. 135-144, 1984.
3. E. F. Codd, *Cellular Automata*, Academic Press, New York, 1968.
4. R. A. Freitas and W. P. Gilbreath, "Advanced Automation for Space Missions", *NASA Report CP-2255*, 1982.
5. J. Byl, "Self-Reproduction in Small Cellular Automata", *Physica 34D*, pp. 295-299, 1989.
6. J. A. Reggia, S. A. Armentrout, H.-H. Chou, Y. Peng, "Simple Systems That Exhibit Self-Directed Replication", *Science*, Vol. 259, pp. 1282-1287, 26 February 1993.