Self-Replication of 3D Universal Structures

André Stauffer, Daniel Mange, Enrico Petraglio and Gianluca Tempesti
Swiss Federal Institute of Technology
Logic Systems Laboratory
CH- 1015 Lausanne, Switzerland
andre.stauffer@epfl.ch

Abstract

After a survey of some realizations of self-replicating machines, this paper presents the construction based self-replication of universal 3D structures. This self-replication process is achieved by translation and transcription of a configuration information in a three-dimensional data and signals cellular automaton (DSCA). The specifications and the design of the basic three-dimensional cell of the automaton results in a new and straightforward methodology for the self-replication of 3D computing machines of any dimensions.

1 Introduction

In the history of non trivial self-replicating machines, there are mainly two different approaches: (1) the self-replication based on inspection, and (2) the self-replication based on construction.

Using inspection in order to implement self-replicating structures was the approach chosen by a few researchers. Morita and Imai present configurations named worms and loops able to self-replicate by using a shape-encoding method. Ibanez and al. describe also a self-inspecting loop capable of self-replication. In these works, the data are only used in an uninterpreted way called transcription. It is accomplished by duplication of signals at the periphery of the structure.

Using construction in order to self-replicate structures was the way von Neumann conceived to solve the problem. It was Langton who realized the first practical implementation of the process. In these approaches, the information is successively used in two different modes, interpreted and uninterpreted. First, in the interpreted mode or translation, the signals are executed as they reach the periphery in order to construct the replicated structure. After, in the uninterpreted mode or transcription, the signals are copied in the replicated structure.

The main goal of this paper is to carry out construction in order to self-replicate universal 3D structures. This self-replication process is achieved in a three-dimensional data and signals cellular automaton (DSCA) [1]. Section 2 presents the specifications of the basic three-dimensional cell of the automaton. The design of this cell is described in Section 3. Section 4 will conclude by opening new avenues based on the self-replication of 3D universal structures.

2 Cell Specifications

The minimal 3D structure is made up of eight cells organized as the $2 \times 2 \times 2$ array represented in Figure 1. Each cell of the array comprises four data expressed in small cubes. As we will see later, the two data to the left are mobile and constitute the genotype of the cell, while the two data to the right are fixed and define the phenotype of the cell. Each of these data is either a code or a flag. The code data is used for configuring the structure whereas the flag data is indispensable for constructing the skeleton of the structure, i.e. defining the information path. Figures 2 to 4 summarizes the 3D graphical representations of the data.

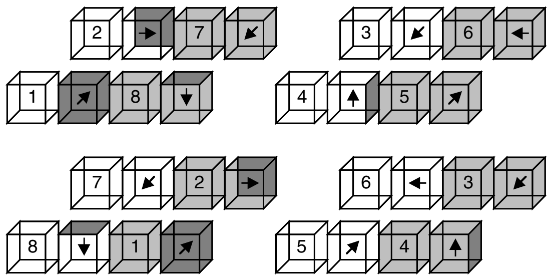

Figure 1. The minimal 3D structure composed of $2 \times 2 \times 2$ cells.

In order to show the construction of our 3D minimal self-replicating structure, we introduce 2D graphical representa-

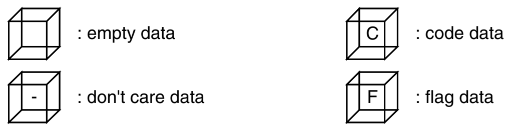

Figure 2. Nature of the data.

-
: mobile data
-
: fixed data

Figure 3. Status of the data.

tions. In Figure 5, the eight cells of the minimal structure are organized as two levels $L = 1$ and $L = 2$ of two rows by two columns. Each cell is able to store in its four memory positions four configuration data. The original configuration is a string of 16 data moving counterclockwise by one data at each time step ($t = 0, 1, 2, ...$).

The 2D graphical representations as well as the hexadecimal representations of the data composing the configuration string are detailed in Figure 6. They are either *empty data* (0), *code data* (from 1 to E) or *flag data* (from 1 to 9 in addition to F). The code data will be used to define the functionality of the structure. The flag data will be used to build the connections between the cells of the structure and to create branches for self-replication. Furthermore, each data is given a status and will eventually be a *mobile data*, indefinitely moving around the structure, or a *fixed data*, definitely trapped in a memory position of a cell.

At each time step, a data of the original configuration string is shifted from right to left and simultaneously stored in the lower leftmost cell (Figure 5). Note that the first, third, ... data of the string (i.e. each odd data) is always a flag F, while the second, fourth, ... data (i.e. each even data) is always a code C. The construction of the structure, i.e. storing the fixed data and defining the paths for mobile

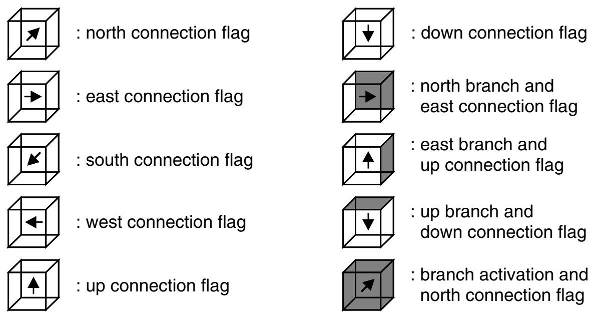
: north connection flag
: east connection flag
: south connection flag
: west connection flag
: up connection flag
: down connection flag
: north branch and east connection flag
: east branch and up connection flag
: up branch and down connection flag
: branch activation and north connection flag

Figure 4. Detailed flag data.

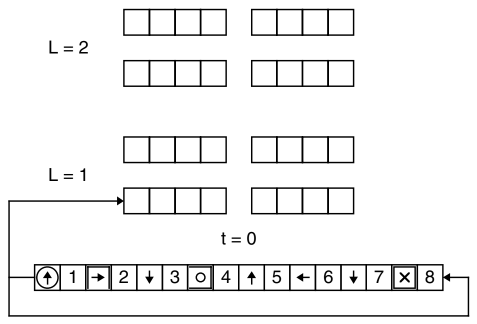

Figure 5. The minimal structure ($2 \times 2 \times 2$ cells) with its configuration string at the start ($t = 0$).

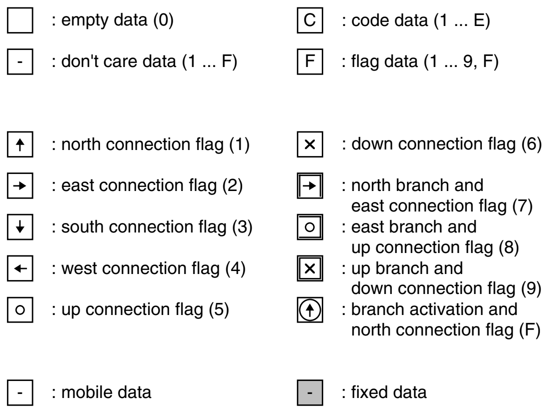
: empty data (0)
C
: code data (1 ... E)
-
: don't care data (1 ... F)
F
: flag data (1 ... 9, F)
: north connection flag (1)
: down connection flag (6)
: east connection flag (2)
: north branch and east connection flag (7)
: south connection flag (3)
: east branch and up connection flag (8)
: west connection flag (4)
: up branch and down connection flag (9)
: up connection flag (5)
: branch activation and north connection flag (F)
-
: mobile data
-
: fixed data

Figure 6. 2D graphical and hexadecimal representations of the data.

data, depends on two major patterns (Figure 7).

- If the two, three or four rightmost memory positions of a cell are empty (blank squares), the data are shifted by one position to the right (shift data).
- If the rightmost memory position is empty, the data are shifted by one position to the right (load data). In this situation, the rightmost F' and C' data are trapped in the cell (fixed data), and a new connection is established from the second leftmost position toward the northward, eastward, southward, westward, upward or downward cell, depending on the fixed flag information (F' = 1 or F, 2 or 7, 3, 4, 5 or 8, 6 or 9).

Applying the memory patterns of Figure 7 to our original configuration string, we get two data trapped in a cell and a new connection toward another cell of the structure every four time steps (Figure 8). At time t = 32, 32 data,

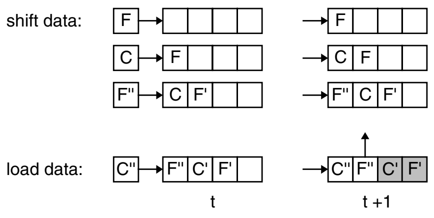

Figure 7. The two memory patterns for constructing a structure.

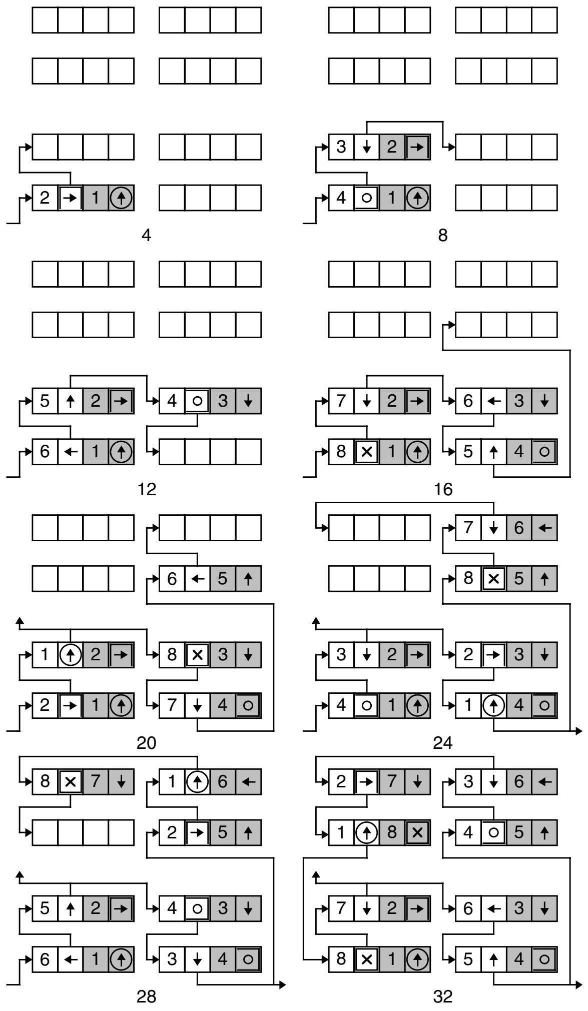

Figure 8. Constructing the minimal structure ($t = 4$: north path, $t = 8$: east path, $t = 12$: south path, $t = 16$: up path, $t = 20$: north path ($L = 2$) and north branch ($L = 1$), $t = 24$: west path ($L = 2$) and east branch ($L = 1$), $t = 28$: south path, $t = 32$: down path and structure completion).

i.e. twice the contents of the original configuration, have been stored in the 32 memory positions of the structure. 16 data are fixed data, forming the phenotype of the final structure, and the 16 remaining ones are mobile data, composing a copy of the original configuration, i.e. the genotype. Both *translation* (i.e. construction of the structure) and *transcription* (i.e. copy of the configuration) have been therefore achieved.

In order to self-replicate, the original structure is able to trigger the construction of three copies, nothward, eastward and upward. At time $t = 19$, the pattern of data initiates the construction of the northward structure. In this pattern, the lower level upper leftmost cell is characterized by two specific flags, i.e. a fixed flag indicating a north branch (F = 7) and the branch activation flag (F = F). This pattern is visible in Figure 9 (northward signal, third row). The new path to the northward structure starts from the second leftmost memory position (Figure 8). At time $t = 23$ and $t = 47$, the patterns corresponding to the third row of the eastward and upward signals in Figure 9 initiate self-replication of the structure to the east and to the top respectively. The other patterns are needed for constructing the inner paths of the structure.

The self-replicating structure in Figure 10 is an example of a non minimal four-column, three-rows and three-level structure. All the non minimal structures can be realized according to this implementation which keeps the number of column even in order to properly close the data path. These non minimal structures involve a new flag (Figure 11) and two more construction patterns (Figure 12).

3 Cell Design

Data and signals cellular automata (DSCA) were originally conceived to provide a formal framework for designing growing structures [1]. Such an automaton is made up of an array of cells, each of which is implemented as a digital system processing both data and signals in discrete time steps. The cellular array (grid) is n-dimensional, where $n = 1, 2, 3$ is used in practice.

In our 3D structure application, the basic cell of the DSCA is three-dimensional and works with the northward (N), eastward (E), southward (S), westward (W), upward (U) and downward (D) directed data (D) and signals (S) (Figure 13). The cell computes its digital outputs O from its digital inputs I. These data and signals outputs are not necessarily identical for all the neighboring cells.

Each cell of the automaton is designed as a digital sys-

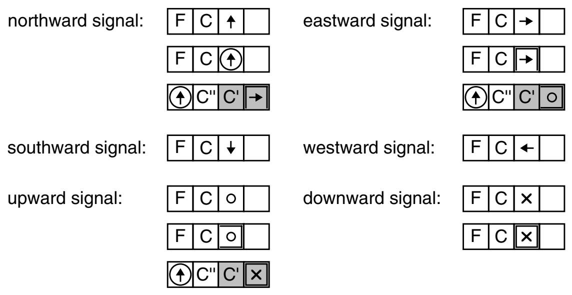

Figure 9. Patterns of data triggering the paths to the north, east, south, west, up and down cells.

tem, resulting from the interconnection of a data processing unit PU and a control unit CU (Figure 14). In this digital system, the processing unit handles the data. It is made up of input selectors, data registers, and output buffers. The control unit of the digital system computes the signals. It combines input encoders, control registers, and output generators.

The processing unit of our cell performs the shift data and load data operations represented in Figure 7. The control unit of our cell produces the signals implied in the construction of the connections between the cells of the structure (Figures 9 and 12). The resources needed in order to

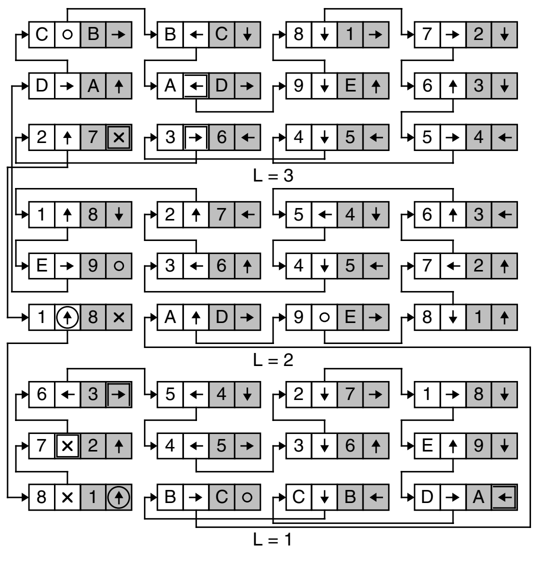

Figure 10. Example of a non minimal structure ($4 \times 3 \times 3$ cells).

: east branch and west connection flag (A)

Figure 11. 2D graphical and hexadecimal representations of the additional data.

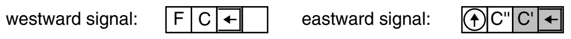

Figure 12. Additional patterns of data triggering the paths to the west cell and the east replicated structure.

do it define the architecture of the cell (Figure 15). The processing unit involves the following ones:

- Two 4-bit genotypic registers GA3:0 and GB3:0 for the propagation of the configuration data.
- Two 4-bit phenotypic registers PA3:0 and PB3:0 for the memorization of the configuration data.
- A 6-input multiplexer DIMUX for the selection of one of the six data input lines, $NDI3:0$, $EDI3:0$, $SDI3:0$, $WDI3:0$, $UDI3:0$, or $DDI3:0$.
- A buffer DOBUF to enable the data output $DO3:0$.

The control unit consists of three resources:

- A 4-bit data input register I3:0 for the memorization of the selection operated by the multiplexer DIMUX.
- A signal inputs SI encoder ENC.
- A signal outputs SO generator GEN.

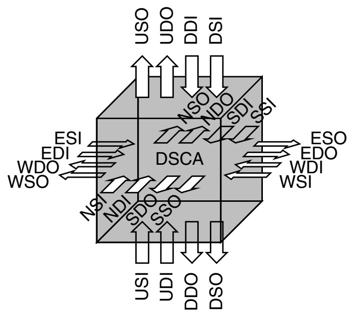

Figure 13. Basic cell of the three-dimensional seven-neighbor DSCA.

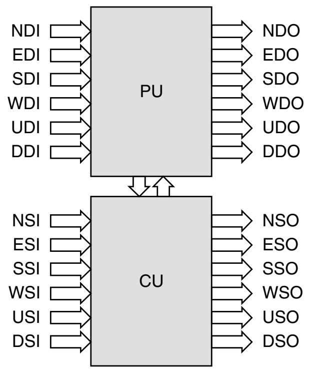

Figure 14. Processing unit PU and control unit CU of the automaton cell.

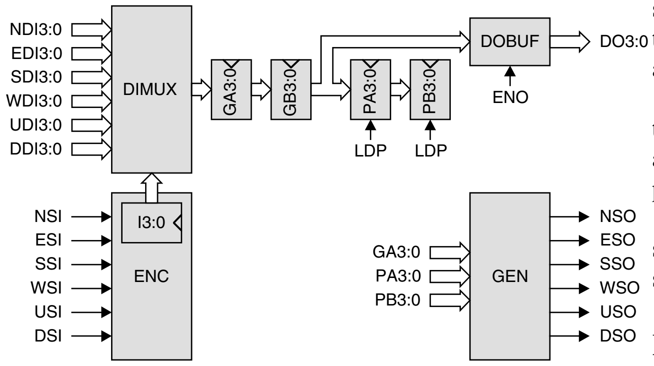

Figure 15. Detailed architecture of the 3D structure cell.

The genotypic registers GA3:0 and GB3:0 always propagate the data DI selected by the multiplexer DIMUX , while the phenotypic registers PA3:0 and PB3:0 perform a hold operation or a load operation according to the control variable LDP. When equal to 1 this control variable expresses that the phenotypic register PB3:0 contains an empty data.

The selection realized by the data input multiplexer DIMUX is specified by the data input register I3:0.

The operations of the data output buffer DOBUF implies the control variable ENO. When equal to 1 this control variable expresses that the phenotypic register PB3:0 contains a flag data.

The data input register I3:0 performs a load operation every time that there is at least one input signal equal to 1.

The encoder ENC operates a priority coding of the input signals.

The generator GEN implements the output signals implied in the construction of the connections according to the patterns of Figures 9 and 12.

4 Conclusion

Several years before the publication of the historical paper by Crick and Watson revealing the existence and the detailed architecture of the DNA double helix, von Neumann was already able to point out that self-replication was a two-mode process able to both interpret (translation mode) and copy (transcription mode) a one-dimensional description, the configuration string. Self-replication will allow not only to grow, but also to repair complete 3D structures. Self-replication is now considered as a central mechanism indispensable for those circuits which will be implemented through the nascent field of nanotechnologies [2].

A first field of application of the self-replication of 3D structures is quite naturally the classical self-replicating automata, such as three-dimensional reversible automata or asynchronous cellular automata.

A second, and possibly more important field of application is Embryonics, where artificial multicellular organisms are based on the growth of a cluster of cells, themselves produced by cellular division [3] [4].

Other possible open avenues are about the evolution of such structures and/or their capability of carrying out massive parallel computation.

References

[1] A. Stauffer and M. Sipper. Data and signals: A new kind of cellular automaton for growing systems. In J. Lohn, R. Zebulum, J. Steincamp, D. Keymeulen, A. Stoica, and M. I. Ferguson (Eds.), Proceedings of the 2003 NASA/DOD Conference on Evolvable Hardware, pp.235–241, IEEE Computer Society, Los Alamitos CA, 2003.

[2] K. E. Drexler. Nanosystems: Molecular Machinery, Manufacturing, and Computation. John Wiley, New York, 1992.

[3] N. J. Macias and L. J. K. Durbeck. Self-assembling circuits with autonomous fault handling. In A. Stoica, J. Lohn, R. Katz, D. Keymeulen and, R. S. Zebulum (Eds.), Proceedings of the 2002 NASA/DOD Workshop on Evolvable Hardware, pp.46–55, IEEE Computer Society, Los Alamitos CA, 2002.

[4] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward robust integrated circuits: The Embryonics approach. Proceedings of the IEEE, vol.88, no.4, pp.516–541, April 2000.