Fault-Tolerant Structures: Towards Robust Self-Replication in a Probabilistic Environment

Daniel C. Bünzli and Mathieu S. Capcarrere[1]

Logic Systems Laboratory
Lausanne Swiss Federal Institute of Technology
CH-1015 INN-Ecublens
{Buenzli.Daniel,Capcarrere.Mathieu}@epfl.ch
http://lslwww.epfl.ch

Abstract. Self-replicating structures in cellular automata have been extensively studied in the past as models of Artificial Life. However, CAs, unlike the biological cellular model, are very brittle: any faulty cell usually leads to the complete destruction of any emerging structures. In this paper, we propose a method, inspired by error-correcting-code theory, to develop fault-resistant rules at, almost, no extra cost. We then propose fault-tolerant substructures necessary to future fault-tolerant self-replicating structures.

1 Introduction

Self-replicating structures in cellular automata have generated a large quantity of papers [10]. In the Artificial Life field, most of these have been devoted to the study of the self-replicating process in itself, as a model of one of life primary property. The underlying motivation of such researches could be expressed as a question: What are the minimal information processing principles involved in self-replication ?

However, if life was an awe inspiring model, simplification was a prime constraint in all these studies, not only for obvious practical reasons, but also as a guide to extract the quintessential ideas behind self-replication. We believe that an always present quality in natural systems was omitted more for the former reasons than the latter principle: *robustness*.

Cellular automata are discrete time and space grid in which all cells update synchronously according to a deterministic rule. No faults ever occurs in the transition rule nor on the cell current state. Nevertheless if we are to look at real biology, it appears at all levels that either insensitiveness to faults or self-repair is a prime condition of survival, the environment being extremely noisy and uncertain. Fault resistant cellular automata designed in the past[2,3,4] are based on hierarchical constructions, i.e., an alteration of the inner architecture of the CA. In this paper, we rather propose to develop fault-tolerant transition rules, keeping the classical CA unchanged. Thereby we create structures that

J. Kelemen and P. Sosík (Eds.): ECAL 2001, LNAI 2159, pp. 90–99, 2001.

are fault-tolerant in their functioning itself, at a minimal cost in terms of space, time or memory.

In section 2, we will accurately define what we mean by a faulty environment, and evaluate its impact on classical self-replicating structures, such as Langton's loop. In the following section, after a short presentation of the error correcting code theory, we will use it to propose a general model of fault-resistant rules. Finally, in section 4, we first present general practical constraints, and then propose two essential fault-resistant substructures for self-replication: a data pipe and a signal duplicator, showing the specificities of such realizations situations compared to the theoretical model.

2 A Non-deterministic Cellular Automata Space

The idea of fault-tolerance is rather vague and our intention in this section is *not* to state any definitive definition. Rather, we propose here, first of all, to delimit the scope of validity of our work, and, secondly, to briefly argue of its validity as a practical environment. Besides, we will show that this noisy environment is more than enough to irrevocably perturbate and thus destroy any of the most famous self-replicating structures.

2.1 Definition of the Faulty Environment

Formally cellular automata are d-dimensional discrete space, in which each point, called a cell, can take a finite number of state q. Each cell updates its own states according to the states of a fixed neighborhood of size n following a deterministic rule. A CA is called uniform when this rule is the same for every cell. Hence a CA may be defined by its transition rule S:

$$S : q^n \to q$$

In CAs, faults may basically occur at two levels : the synchronization (time) and the cell transition rule (space). We do not cater for the former problem in this article, though we argue in a paper to come [1], that a simple time-stamping method exploiting its inherent parallelism correct efficiently all synchronization faults. As for the latter problem, which could be further divided into reading and writing errors, we model it as a non-deterministic rule. More precisely, there is a probability of faults p_f, such that any given cell will follow the transition rule with probability $(1 - p_f)$, and take any state with probability p_f. This model, though apparently catering only for writing errors, do simulate perfectly reading errors. One may object that we did not mention the major fault problems when a cell simply stops functioning at all. This is not our purpose to treat that kind of 'hardware' problem here, but we may say that our model could be implemented over an embryonics tissue[6] which deals with this kind of malfunction. Besides, one may note that such a permanent failure is not, in itself, fatal to our system, but only weakens all intersecting neighborhoods.

2.2 Classical Self-Replicating Structures Behavior

In this section, we show the brittleness of some famous and less famous classical self-relicating structures. We first consider the Langton's loop[5], we then move on to the more complex Tempesti's loop[11] to finish with one of the simplest self-replicating loop that was described by Chou and Reggia[8]. We do not consider the seminal work of Von Neumann[7] for obvious practical reasons. However one may note that the very accurate nature of that latter work leads to a "natural" brittleness.

Applying this model to the landmark self-replicating structures of Langton demonstrates a high sensitivity to failure. Following our model of noisy environment described above, a probability of fault p_f as low as 0.0001 shows a complete degradation of the self-replication function. Tempesti's structure shows exactly the same type of failure for an even lower p_f. The main cause of this total fault intolerance is due to what we may call irreducible garbage remains. Both these loops have as default rule, no change. When confronted to an unexpected neighborhood, i.e., when the fault only generates surprise and not confusion, the cell remains in its previous state. Then garbage that appears in the void (in the middle of a quiescent zone) is irreducible and remains forever, or at least, and that is the source of this higher-than-expected sensitivity to faults, until the loop meets it for its inevitable loss.

The Chou and Reggia loop[8] (p.286, fig. 1g), begins to show significant degradation with a probability p_f of 0.001. This relative resistance is only due to its small size (3x3), and any fault affecting the loop almost inevitably leads to its complete destruction. One may note that this loop suffers, as the preceding examples, from *irreducible remains*.

It is clear that these rules where not designed to be fault-tolerant and thus their size is their only means of defense. Effectively, their probability of failure is directly proportional to their size, and may be approximated by $(1-(1-p_f)^{size})$. Of course this probability to be accurate should be augmented by the number of neighboring cell, and, as we saw, by the probability of encountering *irreducible remains*.

We now present how to design fault-tolerant structures.

3 Fault-Resistant Rules

In a noisy environment, as defined earlier, a cell cannot trust entirely its neighborhood to define its future state. In other works [4,2,3], where the aim was to design generic faultless cellular automata, a hierarchical construction was used, e.g., a meta-cell of dimension 2 creates a fault-tolerant cell, to the cost of a lot more complex architecture and greater computation time. However our approach here is *not* to create a fault-tolerant CA architecture, but rather to design the rules of a classical CAs so that the emergent, global behavior is preserved. That latter choice allows both a reduced computation time and lesser memory space than the former, the cost being a resulting CA specific to one task. In this section

we first briefly present some element of error correcting code theory, on which we base our design of fault-tolerant rule presented in section 3.2.

3.1 Error-Correcting Code

The idea behind the error correcting code theory is to code a message so that when sent over a noisy transmission channel, at reception, one can detect and correct potential errors through decoding, and thus reconstruct the original message.

Formally, let Σ be an alphabet of q symbols, let a code $\mathcal{C}$ be a collection of sequence c_i of exactly n symbols of Σ, and let $d(c_i, c_j)$ be the Hamming distance between c_i and c_j. We call $d(\mathcal{C})$ the minimal distance of a code $\mathcal{C}$ which is defined as $\min_{i \neq j} d(c_i, c_j), c_i, c_j \in \mathcal{C}$. Then the idea of error-correcting code, introduced by Shannon[9], is to decode, on reception, the word received by the closest word belonging to $\mathcal{C}$. Using this simple strategy with a code $\mathcal{C}$ of minimal distance d, allows the correction of up to $(d-1)/2$ errors.

In this theory, one may see that the number of word in $\mathcal{C}$, M, and the minimal distance d, plays against one another for a fixed n and q. To avoid a waste of memory space, in our fault-tolerant rules developed in the next subsection, it would be useful to maximize M for a given d. However it is not always possible to determine it *a priori*, but for $d = 2e + 1$, so that we can correct e errors, this maximum M, $A_q(n, d)$, is bounded. The lower and upper bounds known as Gilbert-Varshamov[12] bounds, are:

$$\frac{q^n}{\sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i} \leq A_q(n,d) \leq \frac{q^n}{\sum_{k=0}^{e} \binom{n}{k} (q-1)^k} \tag{1}$$

3.2 Theoretical Aspect of Fault-Resistant Rules

A rule r in a 2-dimensional cellular automata is the mapping between a neighborhood, v, and a future state, c^+, that, if we use Moore neighborhood, may be written as $(v_1, ..., v_9 \rightarrow c^+)$. In the noisy environment, as defined in section 2.1, the values $v_1, ..., v_9$ may be corrupted, thereby inducing in most cases, a corrupted value of c^+, propagating in time the original error. The idea behind this paper is to see c^+ as the message to be transmitted over a noisy channel, and $v_1, ..., v_9$ as the encoding of the message. Then it becomes clear that if this encoding does respect the error correcting code theory constraints exposed above then, even if the neighborhood values are corrupted, it will still be decoded into the correct c^+ future state and the original errors will not last further than one time step.

Of course, the constraints will depend on the number of errors, e, we want to be able to correct for a given neighborhood. We define $(v_1, ..., v_9 \rightarrow c^+, e)$ to be the transition rule r resisting e errors. In consequence, we also define $V(r)$ to be the set of rules co-defined by r, $V(r) = ((\overline{x} \rightarrow c^+) \mid d(\overline{x}, (v_1, ..., v_9)) \leq e)$. Then

it appears clearly that if each rule of the co-defined set of rules is at a minimal hamming distance of 1 of any other rule of the co-defined set of rules, that is, if each main rule is at minimal hamming distance of $2e + 1$ of any other main rule, then, reinterpreting the result of Shannon above, we can guarantee that the CA thus defined will be able to function correctly providing that at most e errors happen at any one time step in each neighborhood. As one may note, the size of $V(r)$ is rather large. More precisely, for a neighborhood of size n, and an alphabet of q symbols we have:

$$|V(r)| = \sum_{k=0}^{e} \binom{n}{k} (q-1)^k \tag{2}$$

A *conflict* appears when the intersection between every co-defined set of rules is not empty. For instance, for a von Neumann neighborhood, $r_1 = (03234 \to 1, 1)$ and $r_2 = (03034 \to 2, 1)$ are in major conflict as the neighborhood (03234) lies both in $V(r_1)$ and $V(r_2)$. To avoid wasting too many possible different neighborhoods, we distinguish *major* and *minor* conflicts. A major conflict appears when the future state of the conflicting rules are different. However minor conflicts, when future states of the conflicting rules are identical, are reasonable as it does not prevent error correction. Thus, we have the following reinterpreted Shannon's theorem:

Theorem 1. *Let $\mathcal{R}$ be all the transition rules of a CA* `C`*, then if we have:*

$$\forall_{i \neq j}(r_i \to c_i^+), (r_j \to c_j^+) \in \mathcal{R}, \quad d(r_i, r_j) \geq 2e + 1 \textbf{ or } c_i^+ = c_j^+$$

we can guarantee that the CA `C` *will correct up to e errors occurring at any one time step in any neighborhoods.*

We have now defined theoretically how to conceive a fault-tolerant structures on a classical CA. In the following section, we study the practical application of the above defined constraints to peculiar substructures of interest for self-replication. On this more practical aspect of the question, it would be interesting to know how many different rules are available for use. If we do not take into account the distinction of minor and major conflict, and as we need all the rules to be at a hamming distance of $d = 2e+1$ to guarantee distinct co-defined set of rules, the number of rules available is given by the bounds of Gilbert-Varshamov (eq. 1). Nevertheless these bounds are rather large, and do not include the large advantages brought by the minor conflict exception. In any case the waste of possible rule is usually quite important self-replicating structures. If we look at classical structures, such as Langton's loop, the number of used rule compared to the number of rules available (namely q^n), is well within the Gilbert-Varshamov bounds for a minimal number of errors $e = 1$. On the other hand, even for $e = 1$, if the probability of failure of a cell is p_f, and the size of the neighborhood is n, then the probability of failure of our structure, whatever the size of this structure, is reduced, roughly, to $1 - ((1 - p_f)^9) - 9 * p_f(1 - p_f)^8$.

4 Fault-Resistant Structures

In this section, we propose to apply practically the theoretical ideas put forward in the previous section. We will only consider rules that can correct up to 1 error in their neighborhood, as it seems a reasonable compromise between the fault-tolerance capabilities and the implied "waste" of rules. In the same way of thinking, we will only consider two-dimensional Moore neighborhood, as it provides a wealth of rules compared to the von Neumann neighborhood at no real additional cost in terms of computational time, memory space or physical complexity.

The aim of this research is to develop a fault-tolerant self-replicating structure. Firstly, we will study some common problems and desirable properties to construct such structures. Secondly, in section 4.2, we show how to practically construct a fault-tolerant data pipe, a structure essential to convey information. Finally, in section 4.3, we propose a fault-tolerant signal duplicator, central to the whole idea of self-replication.

4.1 Properties, Methods, and Problems

Our CA's structure are quite classically defined as an "active" structure in a pool, possibly infinite, of quiescent cells (henceforth represented by the '.' symbol). Hence the first necessary rule is $r_q = (..... \rightarrow ., 1)$. Its co-defined set of rules $V(r_q)$ defines all the rules with one active cell. It has the advantage to cover what we called in section 2.2, the irreducible remains. Effectively, any fault creating a single active cell in a quiescent neighborhood will be reverted to the quiescent state at the next time step. This eliminates the main problem encountered by the non fault-tolerant structures. Although the encounter of our fault-tolerant rules with the remains would not be as deadly as for the classical loops, this prevents accumulation of errors in the same neighborhood over time. Nevertheless, it is important to note that, consequently to the definition of r_q, we cannot use any rules with less than 3 active cells in its neighborhood. Its co-defined set of rules would otherwise intersects with $V(r_q)$ and create a major conflict unless, of course its future state is '.'. This last constraint which may be seen as a disadvantage in terms of the global size of the structure, is not much of a problem on the point of view of error-correction as the fault-tolerance capabilities of our structures only depend on the size of the neighborhood and not of the global structure.

That latter implied property brings us to a more general remark about the method to use when creating fault-tolerant structures. If we take figure 1, one would be tempted not to define the rule $r_1 = (.....M \rightarrow ., 1)$ arguing that it is covered by $V(r_q)$. However, this would be mistake as $V(r_1) \neq V(r_q)$, we would then lose our guarantee of error correction. Hence, minor conflicts are handy as they allow us to define rules such as r_1 and r_q simultaneously, nevertheless they do not make the definition of r_1 unnecessary if we are insure the fault-tolerance property.

Besides these kind of tricks one should use, we are currently developing a more formal method to 'automatically' create fault-tolerant rules. At the moment, the semi-algorithm goes as follows: When going from the structure from one time step to the next one, we define the rules (with $e = 1$) for each cell in the structure and its neighborhood, going from left to right and top to bottom. After each rule definition, we test for major conflicts. This way to proceed prevents a lot of useless work, but does not discard the design by hand of the initial structure and its successors.

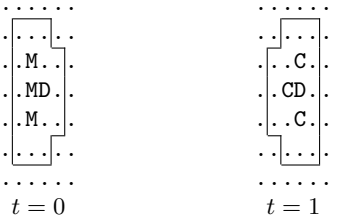

Fig. 1. Structure and its neighborhood

4.2 Safe Data Pipe

Now that we have seen the theoretical aspects and some general practical constraints, we are able to define a fault tolerant data pipe. These kind information transmission structures are essential for self-replicating loops. A typical example of these is given in figure 2.

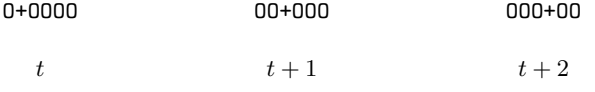

Fig. 2. Usual data pipe

One sees immediately the problem with that kind of

structures. It implies the two following rules, $r_1 = (\texttt{0..0...+.} \rightarrow \texttt{+}, 1)$ and $r_2 = (\texttt{+..0...0.} \rightarrow \texttt{0}, 1)$, which obviously provoke a major conflict. In fact, we remember that, for $e = 1$, each rule must be at a minimal hamming distance of 3. This constraint deals also with the quiescent rule constraint.

We can see in figure 3 an example of a fault-tolerant data pipe for $e = 1$ addressing that constraint. It is able to transmit trains of data. As one may see

two types of data are transmitted in this example :L which uses 3 states to move itself, and + which uses 4. As we will explain later, the 4 states are needed as it is the *head* of the data train.

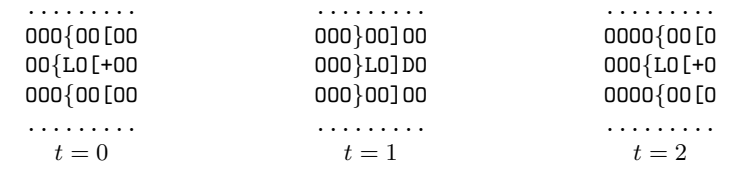

Fig. 3. Fault-tolerant data pipe

We will now have a more detailed look at this fault-tolerant data pipe:

- at $t = 0$, the upper and lower "wings" are necessary to maintain the 'O' state situated at the 'east' of the head of the signal, the '+'state. The state 'O' is designed to conduct the head.
- at $t = 1$, we have a transitory structure. This transition is necessary to maintain the integrity of the pipe. The upper and lower "wings" not moved yet would create a conflict if they were still in the state '['. We would end up with the conflicting transition rules (O..OOD[[.→[, 1) and (O..OOO+[.→O, 1). This transitional step is there to create a diversity in the neighborhoods, thereby suppressing all conflicts.
- at $t = 2$, we get back to the original the structure, the train of signal (that one may define as L+) has moved forward.

As we noted earlier, a supplementary state was required for the head of the signal, the data '+', than for the rest of the signal 'L'. It can be clearly seen why if we imagine the situation without that supplementary state (see figure 4).

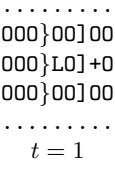

Fig. 4. Problem with the head signal

In this situation we have to define the rule (OOOOOOO+O → O, 1), which then conflict with the rule (OOOOOO[+[→1, 1) defined at the preceding time step. The transitory state 'D' solves that conflict. The neighborhood of L, and any other following signal being more diverse, that supplementary transitory state is useless.

4.3 Signal Duplication

Now that we have, through the detailed definition of a fault-tolerant data pipe, seen the practical aspects of making fault-tolerant structures, we propose in this last section to quickly view a fault-tolerant signal duplicator. This peculiar substructure is always present in self-replicating structures. Actually, its the core of self-replication. The data pipe is there to maintain the structure, the duplicator is there to start a new construction, the replicate. We can check that this rather more complex structures only requires the use of two more states, `C` and `K`, besides the state needed by the data pipe described above.

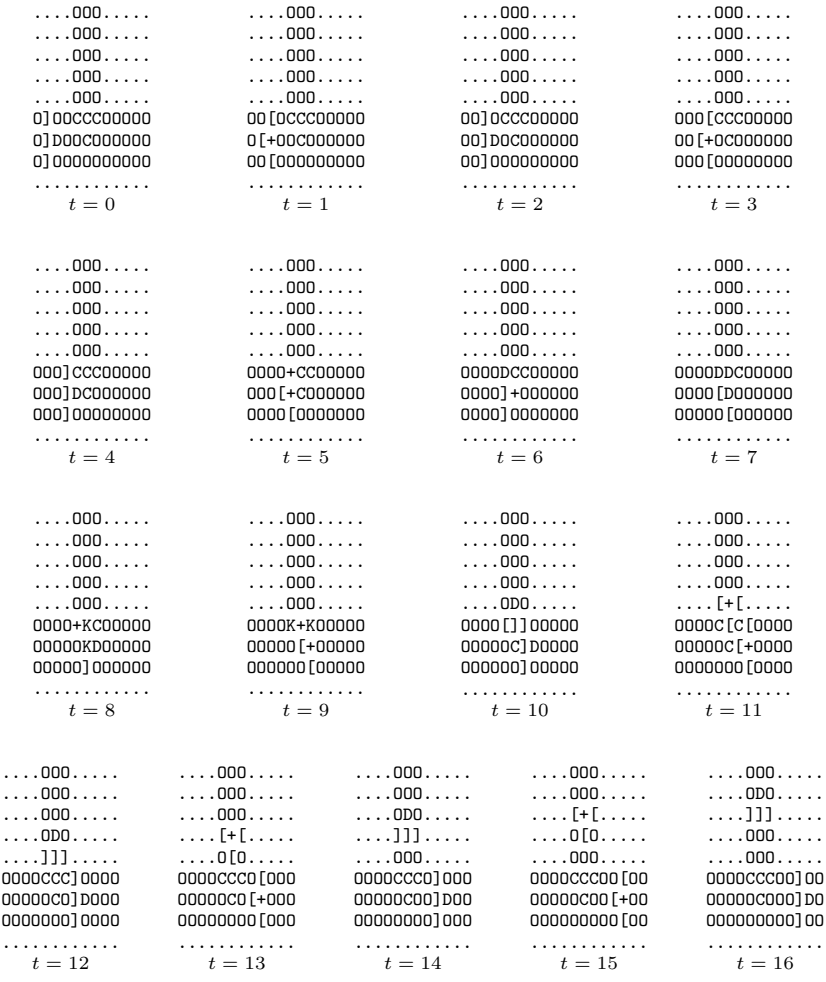

Fig. 5. Duplication of a signal

5 Concluding Remarks

We have shown that it is possible to create fault-tolerant structures by acting on the rules only. While not totally free of costs in terms of memory or computational time, the advantage of this method, in these terms, compared to classical, i.e., hierarchical, fault-tolerant CAs is evident. In terms of error correction capabilities, the rule-based method renders the probability of failure proportionate to the, relatively very small, size of the neighborhood. However, as we have seen in section 4.1, the possibility of algorithmically, and thus automatically, develop fault-resistant rules is still embryonic. Further work will pursue in this direction, thereby eliminating the main drawback of our method. Effectively, at the moment we only have task specific hand-made rules.

The two substructures presented are at the core of self-replicating loops, and we should be able in the near future to present a complete self-replicating structure resistant to a noisy environment. We believe that this will be attained with no more complexity, in terms of the number of state or the CA architecture, than most classical loops presented to date.

References

1. Mathieu S. Capcarrere. Asynchronous cellular automata: Efficient designed and evolved solutions for the synchronization and density problems. *Submitted*, 2001.
2. Peter Gács. Self-correcting two-dimensional arrays. In Silvio Micali, editor, *Randomness in computation*, volume 5 of *Advances in Computing Research*, pages 223–326, Greenwich, Conn, 1989. JAI Press.
3. Peter Gács. Reliable cellular automata with self-organization. In *Proceedings of the 38th IEEE Symposium on the Foundation of Computer Science*, pages 90–99, 1997.
4. Masaretu Harao and Shoici Noguchi. Fault tolerant cellular automata. *Journal of computer and system sciences*, 11:171–185, 1975.
5. Christopher G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.
6. Daniel Mange, Moshe Sipper, Andre Stauffer, and Gianluca Tempesti. Towards robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–541, april 2000.
7. John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A. W. Burks.
8. James A. Reggia, Hui-Sien Chou, and Jason D. Lohn. Self-replicating structures : Evolution, emergence and computation. *Artificial Life*, 4:283–302, 1998.
9. C. E. Shannon. *The mathematical theory of communication*. University of Illinois Press, 1949.
10. Moshe Sipper. Fifty years of research on self-replication: An overview. *Artificial Life*, 4:237–257, 1998.
11. Gianluca Tempesti. A new self-reproducing cellular automaton capable of construction and computation. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd Eur. Conf. on Artificial Life (ECAL95)*, volume 929 of *LNAI*, pages 555–563. Springer-Verlag, 1995.
12. R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk. SSSR*, 117:739–741, 1957.