Bio-Inspired Fault-Tolerance

Elena Dubrova
Royal Institute of Technology
Electrum 229, 164 46 Kista
Sweden
dubrova@kth.se

ABSTRACT

In the last decade, there has been a considerable increase of interest in fault-tolerant computing due to dependability problems related to process scaling, embedded software, and ubiquitous computing. In this paper, we consider an approach to fault-tolerance which is inspired by gene regulatory networks of living cells. Living cells are capable of maintaining their functionality under a variety of genetic changes and external perturbations. They have natural self-healing, self-maintaining, self-replicating and self-assembling mechanisms. The fault-tolerance of living cells is due to the intrinsic robustness of attractors' landscapes of their gene regulatory networks. Previously, we introduced a technique which exploits the stability of attractors to achieve a fault-tolerant computation. In this paper, we evaluate this technique on the example of a gene regulatory network model of Arabidopsis thaliana and show that it can tolerate 70% single-point mutations in the outputs of the defining tables of gene functions.

Keywords

Fault-tolerance, gene regulatory network, Boolean network, attractor

1. INTRODUCTION

Fault tolerance is the ability of a system to continue performing its intended function in the presence of faults [30]. In a broad sense, fault tolerance is associated with reliability, successful operation, and the absence of breakdowns. Originally, techniques for fault-tolerance were used to cope with low reliabilities of individual hardware components. Designers of early computing systems used replicated gates and flip-flops to detect faults, or applied voting to correct faults [44, 53]. As semiconductor technology progressed, hardware components became intrinsically more reliable and the need for tolerance of component defect diminished in general purpose applications. Nevertheless, fault tolerance remained an essential attribute of systems used in safety-, mission- and business-critical applications such as military, avionics and aerospace.

During the mid-90, the interest in fault-tolerance resurged considerably. Smaller process sizes lead to an increased likelihood of noise-related faults caused by crosstalk. Lower supply voltage levels, which are used to decrease power dissipation, result in even higher susceptibility to noise [42]. Moreover, denser feature sizes considerably increase the probability of *soft errors* [13] (also called *transient faults*, or *glitches* [3]) caused by cosmic rays and alpha particles. Shielding and radiation hardening are difficult and not cost-effective for most system designs. Shielding increases the weight and size of the system. Radiation hardening is an expensive process and, when used for a low-volume production, will lead to very costly parts. Therefore, alternative methods for assuring fault-tolerance that do not have these drawbacks are needed for general purpose applications.

Another rapidly growing area demanding fault-tolerance is *embedded software* systems. Since software is inherently more complex and less regular than hardware, achieving sufficient verification coverage is difficult. Conventional testing and debugging methods are inherently slow and unscalable. They are generally incapable of covering functional corner cases or finding hard-to-find bugs that may occur only after hundreds of thousands of cycles (like Intel Pentium FDIV bug [51]). The recent focus on formal methods promises higher coverage, however, due to their large computational complexity they are only applicable for specific applications. As a consequence of incomplete verification, many design faults in software remain undetected, creating a risk of serious accidents like Therac-25 radiation overdoses [36], crash of the British destroyer Sheffield [37], explosion of Ariane 5 [38] and Challenger [2].

Interest in fault tolerance is further boosted by the ongoing shift from the traditional desk-top information processing paradigm, in which a single user engages a single device for a specialized purpose, to *ubiquitous computing*, in which many small, inexpensive networked processing devices are engaged simultaneously and distributed at all scales throughout everyday life [55]. The demand for low-cost, low-area, low-power devices which satisfy high safety and security requirements of ubiquitous computing brings a need for unconventional approaches to fault-tolerance.

In this paper, we discuss a possibility of finding new ways to achieve fault tolerance by investigating principles used in the creation of living cells. A living cell can be considered as a molecular computer that configures itself as part of the execution of its code. Cells can maintain their performance

BIONETICS '08, November 25-28, 2008, Hyogo, Japan

under a broad range of random perturbations, varying from temporary chemical or physical changes in the environment to permanent genetic mutations. The core signaling network of a cell is the gene regulatory network. The fault-tolerance of living cells is due to the intrinsic robustness of attractors' landscape of their gene regulatory network. The attractors' landscape is determined by the dynamical phase in which the networks operate, which is, in turn, determined by the allocation of redundancy in the network. By understanding the principles of redundancy allocation at the genetic level, we may find ways to build fault-tolerant chips that can self-heal, self-maintain, self-replicate and self-assemble.

In [22] we introduced a computational scheme which exploits the stability of attractors to achieve fault-tolerance in a non-traditional way. In this scheme, the states of a Boolean network represent variables of the computed function, and attractors represent function's values. In this paper, we evaluate the robustness of this computational scheme on the example of a gene regulatory network model of Arabidopsis thaliana and show that it can tolerate 70% single-point mutations in the outputs of the defining tables of gene functions.

The paper is organized as follows. In Section 2, we give a brief introduction to gene regulatory networks. In Section 3, we describe an abstract model of gene regulatory networks called synchronous Boolean networks. In Section 4, we summarize a Boolean network-based computation scheme presented in [22]. In Section 5 we describe an algorithm for checking equivalence of two functions represented by different Boolean networks and in Section 6 apply this algorithm to evaluate the robustness of the computation scheme from [22]. Section 7 analyzes the relation between redundancy allocation and fault-tolerance. Section 8 concludes the paper and discusses open problems.

2. GENE REGULATORY NETWORKS

The *Gene Regulatory Network* (GRN) is one of the most important signaling networks in living cells [7]. It is composed of the interactions of proteins with the genome. The major discovery related to GRNs was made in 1961 by French biologists François Jacob and Jacques Monod [29]. They found that a small fraction of the thousands of genes in the DNA molecule acts as tiny "switches". By exposing a cell to a certain hormone, these switches can be turned "on" or "off". The activated genes send chemical signals to other genes which, in turn, get either activated or repressed. The signals propagate along the cell until it settles down into a stable pattern.

Jacob and Monod's discovery showed that the DNA is not just a blueprint for the cell, but rather an automaton which allows for the creation of different types of cells. It answered the long open question of how one fertilized egg cell can differentiate itself into brain cells, lung cells, muscle cells, and other types of cells that form a newborn baby. Each kind of cells corresponds to a different pattern of activated genes in the automaton.

Jacob and Monod introduced the first model of the GRN described by a system of differential equations for the activation and deactivation of the set of genes controlling the transport and metabolism of lactose in *E. coli*. Since then, this little genetic circuit, known as the *lac operon*, has been a prototype for the modeling of the GRN. It is a point of view of many biologists that the only valid approach for the modeling of the GRN is through differential equations.

In 1969 Stuart Kauffman proposed an alternative model of the GRN, called Boolean network [31]. In this model, one is interested in the state of expression of the genes rather than in the concentration of their products. The genome is represented by a set of Boolean variables which are related to each other through some logical rules. For many years, the Kauffman model was considered as an over-simplification of the GRN. Most did not believe that the Boolean approach can yield accurate descriptions of real biological systems.

However, recently it turned out that Kauffman's model is much more powerful than it was originally thought. Experimental and numerical evidence have shown that gene expression profiles of real organisms can be recovered by using the Boolean approach [43, 5, 23]. Kauffman's hypothesis stating that dynamical attractors of the GRN correspond to cell types have been investigated and confirmed experimentally [28, 27]. These results show that the Boolean network model indeed captures the essential aspects of the interactions between the genes.

Many other network models of the GRN have been proposed (see [46] for an overview). Although the details of the network's dynamics might change from continuous in one model to discrete in the other, the general properties of the dynamics appear to be model independent. It has been demonstrated that continuous and discrete descriptions of the GRN exhibit similar dynamical properties under very general conditions [50, 17]. For example, a Boolean network of the *Drosophila melanogaster* [6] has been shown to recover the same patterns for segment polarity genes as those recovered by a continuous model [52]. Discrete models are further justified because recent experimental evidence suggests that, at the individual cell level, gene expression is digital and stochastic rather than continuous [16].

Apart of the simplifying assumption regarding the discrete states of a gene, the Boolean network model assumes the synchronous updating. It has been demonstrated that synchronous and asynchronous updating schemes yield very similar critical stability values, meaning that the transition between the dynamical phases does not depend on the updating scheme [25].

Lastly, the restriction to only two binary states has also been shown justifiable. In [16], the original ternary-state network model of *Arabidopsis thaliana* [23] has been translated to a binary-state one. It has been shown that the Boolean model reaches the same number and type of attractors as the ones reached by the original model. It also responds to perturbations in a qualitatively identical manner to the ternary one. These results suggest that binary states are sufficient to capture the dynamical features of the GRN.

3. BOOLEAN NETWORKS

In the *synchronous Boolean network model* of the GRN [31], every gene is represented by a vertex in a directed graph with an associated Boolean variable x_i that can take two values: $x_i = 1$ if the gene is expressed and $x_i = 0$ otherwise. The genome is represented by a set of n variables, $x_1, x_2, ..., x_n$. An edge from one vertex to another indicates that the former gene regulates the latter. Time is viewed as proceeding in discrete steps. At each step, the expression of the gene x_i

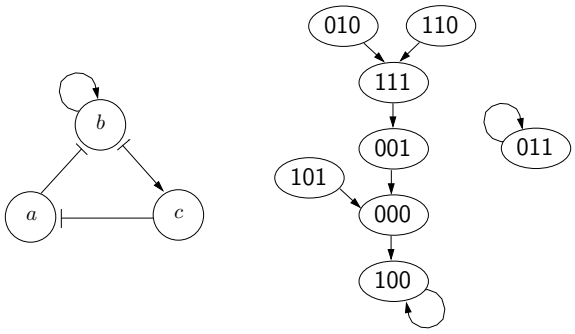

Figure 1: A Boolean network and its state transition graph. Each state is a triple (x_a, x_b, x_c).

changes according to the equation

$$x_i(t+1) = f_i(x_{i_1}(t), x_{i_2}(t), \ldots, x_{i_{k_i}}(t)),$$

where $x_{i_1}, x_{i_2}, \ldots, x_{i_{k_i}}$ are *regulators* of x_i and f_i is a Boolean function which is assigned according to the inhibitory or activatory nature of the regulators. The k_i-tuples of values of the regulators for which $f_i = 1$ are called *activatory* assignments, and those for which $f_i = 0$ are called *inhibitory* assignments.

The *state of the network* is defined as an ordered n-tuple of values of variables $x_1, x_2, ..., x_n$ describing which genes in the network are expressed or not at a particular moment. Since the network is deterministic and finite, any sequence of consecutive states eventually converges to either a single state, or a cycle of states, called *attractor*. The *basin of attraction* of an attractor A is the set of all states from which A can be reached. Kauffman hypothesized that attractors correspond to a combination of gene expressions which specifies a particular cell type or cell fate of an organism [32]. The number of cell types predicted by the Boolean network model agrees well with our current knowledge [28, 33].

An example of a Boolean network is shown in Figure 1. Arrows indicate activatory regulation and blunt-ends represent inhibitory regulation. The following Boolean functions are associated to the vertices:

$$\begin{aligned} f_a &= x'_c \\ f_b &= x_b \cdot (x'_a + x'_c) \\ f_c &= x_b \end{aligned}$$

where "$\cdot$", "$+$" and "$'$" stand for the Boolean AND, OR and NOT, respectively. The state transition graph describing the dynamics of this network is shown on the right-hand side of Figure 1. There are two point (i.e. single-state) attractors, (100) and (011).

The fraction p of activatory assignments of regulators in the entire network, called the *gene expression probability*, is an important parameter which controls the dynamical phase in which the network operates. Dynamics of randomly generated Boolean networks has been extensively studied (see [8] for an overview). It has been shown that, for a given gene expression probability p, the critical value k_c of the mean input degree of vertices at which the transition from ordered to chaotic phase occurs is given by the equation:

$$k_c = \frac{1}{2p(1-p)}. \qquad (1)$$

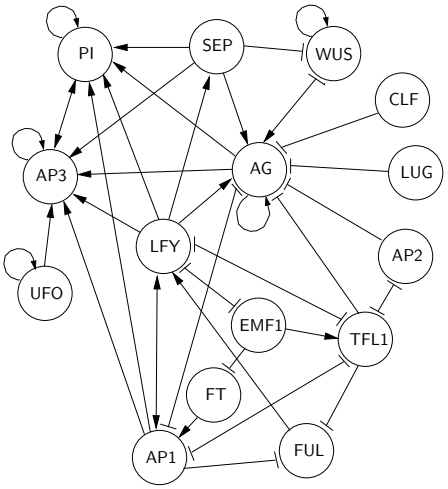

Figure 2: A gene regulatory network model of the wild-type *Arabidopsis thaliana*.

In the infinite size limit, if the mean input degree k of vertices in the network is smaller than k_c, then the network is in the *ordered phase* [24]. If $k > k_c$, then the network is in the *chaotic phase* [39]. If $k = k_c$, then the network is on the *critical line* [11].

Kauffman hypothesized that Boolean networks operating on the critical line are good candidates for the modeling of real GRNs [32]. On one hand, networks in the ordered phase exhibit "frozen" dynamics in which small variations in the initial state of the network typically die out over time. On the other hand, networks operating in the chaotic phase are extremely sensitive to small changes in the initial state, and therefore unstable [26, 39]. A compromise between frozen and chaotic behavior is achieved close to the critical line between the phases. Recent works have shown evidence that GRNs of living cells operate close to the critical line [45, 10]. Critical systems exhibit remarkable properties. For instance, they can integrate, process and transfer information faster and more reliably than non critical systems [47]. They can detect and respond to external stimuli whose intensities span several orders of magnitude, like the brain [35]. These properties are mainly a consequence of the long-range correlations that emerge close to the critical line, producing collective behaviors and coordinated responses of the entire system [10]. Thus, criticality gives a system the ability to respond and adapt to a rapidly changing environment.

Boolean networks have been applied to the problems of cell differentiation [28], immune response [34], evolution [19], and neural networks [9]. They have also been extensively studied by physicists due to their analogy with the disordered systems studied in statistical mechanics, such as the mean field spin glass [18].

4. BOOLEAN NETWORK-BASED COMPUTATIONAL SCHEME

Suppose that we have a Boolean network G with n vertices $v_1, \ldots, v_n$ and m attractors $A_1, \ldots, A_m$. Let the basin of attraction of A_i be denoted by $B(A_i)$. In [22] we introduced the following computational scheme:

AP3	UFO	FUL	FT	AP1	EMF1	LFY	AP2	WUS	AG	LUG	CLF	TFL1	PI	SEP	Cell type
0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	Infl1
0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	Infl2
0	1	0	0	0	1	0	0	1	0	1	1	1	0	0	Infl3
0	0	0	0	0	1	0	0	1	0	1	1	1	0	0	Infl4
1	1	0	1	1	0	1	1	0	0	1	1	0	1	1	Pe1
1	0	0	1	1	0	1	1	0	0	1	1	0	1	1	Pe2
0	0	0	1	1	0	1	1	0	0	1	1	0	0	1	Sep
1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	St1
1	0	1	1	0	0	1	1	0	1	1	1	0	1	1	St2
0	0	1	1	0	0	1	1	0	1	1	1	0	1	1	Car

Table 1: Point attractors of the Boolean network in Figure 2; Infl = inflorescence meristematic cells; Pe = petal primordial cells; Sep = sepal primordial cells; St = stamen primordial cells; Car = carpel primordial cells.

DEFINITION 1. *A Boolean network with n vertices and m attractors $A_1, A_2, \ldots, A_m$ represents a function of type $g : \{0,1\}^n \to \{0,1,\ldots,m-1\}$ which is defined as follows:*

$$g(s_1,\ldots,s_n) = i \quad \textit{if and only if} \quad (s_1,\ldots,s_n) \in B(A_i),$$

for all $(s_1,\ldots,s_n) \in \{0,1\}^n$ and all $i \in \{0,1,\ldots,m-1\}$.

In the definition above, the set of all points of the Boolean space corresponding to the states in the basin of attraction of A_i is mapped to i. Since the basins of attractions partition the Boolean space $\{0,1\}^n$ into m connected components via a dynamic process, the mapping $g : \{0,1\}^n \to \{0,1,\ldots,m-1\}$ is unique up to the permutation of m values of g.

As an example, consider the state transition graph of the Boolean network shown in Figure 1. If we assign the logical 0 to the left-hand side attractor and the logical 1 to the right-hand side attractor, then this network represents the Boolean function $g = x'_a \cdot x_b \cdot x_c$.

The size of the resulting Boolean network representation is linear in the number of variables of the represented function. The maximal number of steps required to compute the value of the function for a given assignment of variables is equal to the longest path to an attractor. Although such a path can potentially be exponential in the number of variables, it is known that Boolean networks on the critical line reach an attractor in a relatively small number of steps [8]. For example, in the Boolean network model of *Arabidopsis thaliana* considered in the next section, an attractor is reached in at most 8 steps [16], while the mean number of steps varies for different attractors from 1.64 to 3.96 [16].

5. EQUIVALENCE CHECKING

In order to evaluate the robustness of the computational scheme given by the Definition 1, we need an algorithm for checking whether a Boolean network after a mutation computes the same function as the original network. In other words, we have to check equivalence of two functions represented by two different Boolean networks. First, we introduce the notion of *state-isomorphism* of state transition graphs.

DEFINITION 2. *Two state transition graphs S_1 and S_2 are state-isomorphic if they have equal number of connected components and, for each component in S_1 there is a component in S_2 which has the same states as S_1.*

As an example, consider the state transition graphs in Figure 3. Both have two components which consist of the same states. So, the graphs are state-isomorphic.

LEMMA 1. *Two Boolean networks represent the same function if and only if their state transition graphs are state-isomorphic.*

The proof follows from the Definitions 1 and 2.

Similarly to the general isomorphism, state-isomorphism is an equivalence relation on state transition graphs which partitions the set of all state transition graphs into equivalence classes. Equivalence classes are related to our study of robustness because a larger equivalence class implies a higher probability of remaining in the same class after the occurrence of a fault.

It is possible to check state-isomorphism using the algorithm for computing attractors which we presented in [22]. This algorithm starts from a randomly selected state and applies forward reachability analysis to find a state in some attractor A_i. Then, using this state as a final state, backward reachability analysis is performed to find the remaining states in the basin of attraction $B(A_i)$. The process is repeated starting from a state not previously visited until the complete state space is covered. We use this algorithm for checking state-isomorphism by running it in parallel on two networks from the same initial state. For each computed attractor A_i, the basins of attractions of two networks are compared for equivalence of states. We use Binary Decision Diagrams (BDDs) [15] for representing the set of states in the basin of attraction, which makes the equivalence checking particularly efficient (a constant-time operation). If the basins are not state-isomorphic, the algorithm terminates. Otherwise, it continues until the complete state space is covered.

6. EVALUATION OF ROBUSTNESS

Living organisms can sustain a wide variety of genetic changes. Gene regulatory networks and metabolic pathways self-organize and re-accommodate to make the organism continue performing under many point mutations, gene duplications and gene deletions [54]. In this section, we demonstrate that the proposed computational scheme inherits the intrinsic robustness of living organisms. As a case study, we take *Arabidopsis thaliana*. Arabidopsis is probably the most studied flower plant [14]. It was the first plant to have its entire genome sequenced in 2000 [4].

Attractor	Basin size
Infl1	512
Infl2	512
Infl3	256
Infl4	256
Sep	448
Pe1	8
Pe2	440
St1	15168
St2	568
Car	14600

Table 2: Size of the basins of attraction of the Boolean network in Figure 2.

Vertex	Number of tolerated faults	Total number of faults
AP3	98	128
UFO	0	2
FUL	4	4
FT	1	2
AP1	2	16
EMF1	0	2
LFY	0	16
AP2	0	2
WUS	0	8
AG	381	512
LUG	0	1
CLF	0	1
TFL1	6	16
PI	56	64
SEP	0	2
total	548	776

Table 3: Results of injecting single faults in each output of all Boolean functions associated to vertices of the Boolean network in Figure 2.

A gene regulatory network model of *Arabidopsis thaliana* floral organ cell fate determination was presented in [16] (see Figure 2). The description of Boolean functions associated to vertices is given in the Appendix. These functions were derived by the authors of [16] from molecular genetic experimental data. The state transition graph of the Boolean network in Figure 2 has 10 point attractors, shown in Table 1. These attractors coincide with the gene expression profiles that have been documented experimentally in the cells of wild-type Arabidopsis. The number of states in each basin of attraction is shown in Table 2. Note that the sizes of the basins of attraction may indicate which genes are critical to attain each cell type [16]. The largest two basins, consisting of 15168 and 14600 states, belong to reproductive organs (stamens and carpels). They are much larger than the basins of the perianth organs (sepals and petals). A larger basin usually implies a more stable attractor, suggesting that the cells of the reproductive organs are more stable than the ones of the perianth organs.

We have tested the robustness of the function $g : \{0,1\}^{15} \rightarrow \{0,1,\ldots,9\}$ represented by the Boolean network in Figure 2 in accordance with the Definition 1 by single random point alterations of the outputs in the defining tables of gene functions. At each run, one alteration was done in one of the outputs of the associated function of a selected network's vertex. After each alteration, the basins of attraction were recomputed and the resulting represented function g compared to the original one using the algorithm described in the previous section. This was repeated for each output of the Boolean functions associated to all vertices of the Boolean network in Figure 2 (776 times in total). The results are summarized in Table 3. As we can see, 228 of the 776 altered networks (29.38%) yielded a different function from the original one. So, 70.62% of single-point mutations in the outputs of the defining tables of gene functions are tolerated.

Total	10	10^2	10^3	10^4	10^5	10^6	10^7
Relevant	5	25	93	270	690	1614	3502

Table 4: Average number of relevant vertices in random Boolean networks on the critical line.

7. REDUNDANCY ALLOCATION

The robustness of biological systems is due to a large percentage of redundancy. Living cells use redundancy in genes as well as redundancy and extensive feedback in regulatory pathways in order to achieve regulatory reliability [41]. Redundancy is believed to act as a protective buffer against genetic damage and harmful mutations, reducing the probability that any single, random offense to the nucleotide sequence will affect the organism [49]. However, the principles of redundancy allocation in GRNs are not well-understood yet. Some studies indicate that the genome is heterogeneous in terms of connectivity of individual genes [40, 16]. A large portion of the genes are *effector* genes that do not directly control the expression of other genes. These effector genes are enslaved by a core of regulatory genes that have direct outputs to the effector genes. Thus, the entire genome of an organism can be seen as a "medusa" network with a regulatory head and many "tentacles" of enslaved effector genes. The dynamical properties of the network are determined by the head of the medusa network, thus the effector genes can be considered redundant. The size of the head of regulatory genes is not known, but is believed to be substantially smaller than the total number of genes of a typical genome. For instance, in *E. coli* virus, the head of the medusa network as computed from the Regulon Data Base consists of less than 80 genes [40]. The total number of genes in *E. coli* genome is 4377 [1].

In the context of Boolean networks, a vertex is considered *redundant* if its removal does not change the attractors of the network [48]. If a vertex is not redundant, it is called *relevant* [12].

Table 1 shows our simulation results for randomly generated Boolean networks on the critical line with homogeneous random topology ($k = 2$ and $p = 0.5$) of sizes from 10 to 10^7 vertices. The first row shows the total number of vertices in the original Boolean network. The second row gives the number of relevant vertices in the reduced network after the removal of the redundant vertices computed as an average for 1000 networks. The redundant vertices were identified using our algorithm RemoveRedundant [21, 20]. RemoveRedundant is an efficient linear-time heuristic. It quickly finds a subset of redundant vertices which are evident from the structure of the network. RemoveRedundant

might miss to identify redundant vertices whose associated functions have constant values due to the correlation of their input variables. Therefore, the actual number of relevant vertices might be smaller than the one shown in the second row of Table 4.

It is well know that a network's robustness is determined not only by the number of redundant vertices, but also by network's topology and by its gene expression probability. The influence of parameters such as input and output degree distributions, type of feedback loops, etc., on networks' fault-tolerance have been extensively studied. However, the specific feature of our approach is that our definition of tolerance is weaker that the usual one. For us, a Boolean network tolerates a fault as long as the basins of attraction of the original and the changed networks are state-isomorphic. This adds another ingredient to the mixture of traditional parameters influencing a network's robustness.

As an example, consider two networks shown in Figure 3. The left-hand side network has the following Boolean functions associated to its vertices:

$$f_a = x'_a + x_b$$
$$f_b = x_a \cdot x_b$$

and the right-hand side network has the functions:

$$f_a = x_a + x'_b$$
$$f_b = x_a \cdot x_b$$

Both networks have the same number of vertices and edges, same gene expressions probabilities, same average number of inputs and outputs, same input and output distributions, and same type of feedback loops. By Definition 1, both networks implement the 2-input Boolean AND. By applying all possible single-point mutations to each output of each associated function (8 in total), we can conclude that the left-hand side network is tolerant (i.e. remains the 2-input Boolean AND) to 4 faults (50%), while the right-hand side network is tolerant to only 2 faults (25%). So, two networks with apparently identical properties have different degrees of robustness.

8. SUMMARY

Traditionally, fault-tolerance of a system is achieved by adding redundant components. However, there are other alternatives. In the computational scheme which we introduced in [22], fault-tolerance is due to the non-uniqueness of paths leading to an attractor. A fault may change a path, but the destination remains the same with a high probability. Therefore, if attractors are stable, the network is able of sustain the majority of faults.

The stability of attractors strongly depends on the dynamical phase in which a network operates. Networks on the critical line have been shown to have stable attractors' landscapes while preserving the ability for evolutionary improvements. From the point of view of the computational scheme from [22], evolvability might be useful for finding a good Boolean network-based representation for a given function. We plan to address this problem in our future work. In general, we can find a Boolean network representing a given Boolean function g by applying traditional sequential circuits synthesis techniques. We can build a state transition graph which partitions the minterms of g into basins of attractions according to Definition 1, and then derive the next-state functions for vertices of the network. However,

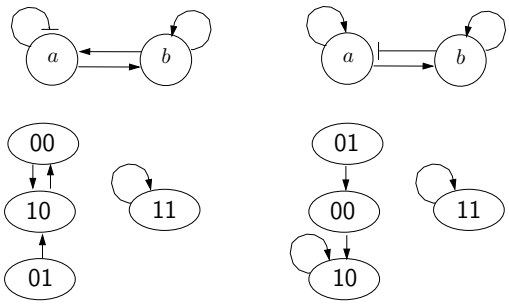

Figure 3: Two networks implementing 2-input Boolean AND and their state transition graphs.

note that functions associated to vertices of a Boolean network model of a GRN are always unate, since a gene regulate another gene in either activatory, or inhibitory way, but not both. Recall that a function $f(x_1, x_2, \ldots, x_n)$ is called *positive unate* in a variable x_i if $f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \geq f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$. Similarly, a function is *negative unate* in x_i if $f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \geq f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$. A function which is either positive or negative unate in every variable is called *unate*. Finding a state transition graph which can be described by unate next-state functions only is an open problem. Selecting a solution which maximizes the robustness of a network is another open problem.

9. REFERENCES

[1] Genome sizes. http://users.rcn.com/jkimball.ma.ultranet/Biology Pages/G/GenomeSizes.html.

[2] Report of the presidential commission on the space shuttle Challenger accident. http://sunnyday.mit.edu/accidents/challenger-table-of-contents.html, 1986.

[3] Terrestrial cosmic rays and soft errors. *IBM Journal of Research and Development*, 40(1), 1996.

[4] The arabidopsis genome initiative (AGI): Analysis of the genome of the flowering plant Arabidopsis thaliana. *Nature*, 408(6814):796–815, 2000.

[5] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J Theor Biol*, 223:1–18, 2002.

[6] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J Theor Biol*, 223:1–18, 2003.

[7] B. Alberts, D. Bray, J. Lewis, M. Ra, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell*. Garland Publishing, New York, 1994.

[8] M. Aldana, S. Coopersmith, and L. P. Kadanoff. Boolean dynamics with random couplings. http://arXiv.org/ abs/adap-org/9305001.

[9] H. Atlan, F. Fogelman-Soulie, J. Salomon, and G. Weisbuch. Random Boolean networks. *Cybernetics and System*, 12:103–121, 2001.

[10] E. Balleze, E. Alvarez-Buylla, A. Chaos, S. Kauffman, I. Shmulevich, and M. Aldana. Critical dynamics in genetic regulatory networks: Examples from four

kingdoms. http://www.fis.unam.mx/ max/English/ paperplos-one-1.pdf.

[11] U. Bastola and G. Parisi. The critical line of Kauffman networks. *J. Theor. Biol.*, 187:117, 1997.

[12] U. Bastola and G. Parisi. Relevant elements, magnetization and dynamic properties in Kauffman networks: a numerical study. *Physica D*, 115:203, 1998.

[13] R. Baumann. Soft errors in advanced comuter systems. *IEEE Design and Test of Comuters*, 22(3):258–266, 2005.

[14] K. D. Birnbaum, D. E. Shasha, J. Y. Wang, J. W. Jung, G. M. Lambert, D. W. Galbraith, and P. N. Benfey. A global view of cellular identity in the Arabidopsis root. In *Proceedings of the International Conference on Arabidopsis Research*, Berlin, Germany, July 2004.

[15] R. Bryant. Graph-based algorithms for Boolean function manipulation. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35:677–691, August 1986.

[16] A. Chaos, M. Aldana, C. Espinosa-Soto, B. G. P. de Leon, A. G. Arroyo, and E. R. Alvarez-Buylla. From genes to flower patterns and evolution: Dynamic models of gene regulatory networks. *Journal of Plant Growth Regulation*, 25(4):278–289, 2006.

[17] M. Chaves, E. D. Sontag, and R. Albert. Methods of robustness analysis for Boolean models of gene control networks. *IEE Proceedings of Systems Biology*, 153:154–167, 2006.

[18] B. Derrida and Y. Pomeau. Random networks of automata: a simple annealed approximation. *Biophys. Lett.*, 1:45, 1986.

[19] E. Dubrova. Modelling of gene regulatory systems by random Boolean networks. *SPIE Bioengineered and Bioinspired Systems II*, 5839:56–65, June 2005.

[20] E. Dubrova and M. Teslenko. Compositional properties of Random Boolean Networks. *Physical Review E*, 71:056116, May 2005.

[21] E. Dubrova, M. Teslenko, and A. Martinelli. Kauffman networks: Analysis and applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 479–484, November 2005.

[22] E. Dubrova, M. Teslenko, and H. Tenhunen. A computational model based on random Boolean networks. In *BIONETICS'2007*, 2007.

[23] C. Espinosa-Soto, P. Padilla-Longoria, and E. R. Alvarez-Buylla. A gene regulatory network model for cell-fate determination during arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *The Plant Cell*, 16:2923–2939, 2004.

[24] H. Flyvbjerg and N. J. Kjaer. Exact solution of Kauffman model with connectivity one. *J. Phys. A: Math. Gen.*, 21:1695, 1988.

[25] C. Gershenson. Updating schemes in random Boolean networks: Do they really matter? In *Proceedings of the Ninth International Conference on the Simulation and Syuntheis of Living Systems*, pages 238–243, 2004.

[26] L. Glass and C. Hill. Ordered and disordered dynamics in random networks. *Europhysics Letter*, 12:599–604, 1998.

[27] S. Huang, G. Eichler, Y. Ber-Yam, and D. E. Ingber. Cell fates as high-dimensional sttractor states of a complex gene regulatory network. *Physical Rev Let*, 94:128701–128704, 2005.

[28] S. Huang and D. E. Ingber. Shape-dependent control of cell growth, differentiation, and apoptosis: Switching between attractors in cell regulatory networks. *Experimental Cell Research*, 261:91–103, 2000.

[29] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–356, 1961.

[30] B. Johnson. *Design and analysis of fault tolerant digital systems*. Addison Wesley Pub, 1989.

[31] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[32] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection of Evolution*. Oxford University Press, Oxford, 1993.

[33] S. A. Kauffman. *Investigations*. Oxford University Press, Oxford, 2002.

[34] S. A. Kauffman and E. D. Weinberger. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology*, 141:211–245, 1989.

[35] O. Kinouchi and M. Copelli. Optimal dynamical range of excitable networks at criticality. *Nat. Phys.*, 2:348–352, 2006.

[36] N. Leveson and C. S. Turner. An investigation of the Therac-25 accidents. In *IEEE Computer*, volume 26, pages 18–41, 1993.

[37] H. Lin. Sheffield hickups caused by software. *Scientific American*, 253(6):48, 1985.

[38] J. L. Lions. Ariane 5 flight 501 failure. http://www.mssl.ucl.ac.uk/www_plasma/missions/cluster/about_cluster/cluster1/ariane5rep.html, 1996.

[39] B. Luque and R. V. Sole. Stable core and chaos control in random Boolean networks. *Journal of Physics A: Mathematical and General*, 31:1533–1537, 1998.

[40] A. Martinez-Antonio and J. Collado-Vides. Identifying global regulators in transcriptional regulatory networks in bacteria. *Curr. Opinion Microbiol*, 6:482/489, 2003.

[41] H. H. McAdams and A. Arkin. It's a noisy business! genetic regulation at the nanomolar scale. *Trends in Genetics*, 15(2):65–69, 1999.

[42] J. D. Meindl, Q. Chen, and J. A. Davis. Limits on silicon nanoelectronics for terascale integration. *Science*, 293:2044–2049, 2001.

[43] L. Mendoza and E. R. Alvarez-Buylla. Genetic regulation of root hair development in Arabidopsis thaliana: A network model. *J Theor Biol*, 204:311–326, 2000.

[44] E. Moore and C. Shannon. Reliable circuits using less reliable relays I. *Journal Franklin Institute*, 263:191–208, September 1956.

[45] M. Nykter, N. D. Price, M. Aldana, S. A. Ramsey, S. A. Kauffman, L. E. Hood, O. Yli-Harja, and I. Shmulevich. Gene expression dynamics in the

macrophage exhibit criticality. *Proc. Natl. Acad. Sci. U.S.A.*, 105 (6):1897–1900, 2008.

[46] T. Schlitt and A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 6:S9, September 2007.

[47] J. P. Sethna and K. A. Dahmen. Crackling noise. *Nature*, 410:242–250, 2001.

[48] J. E. S. Socolar and S. A. Kauffman. Scaling in ordered and critical random Boolean networks. http://arXiv.org/abs/cond-mat/0212306.

[49] J. Suurkula. Over 95 percent of DNA has largely unknown function, 2004. http://www.psrast.org/junkdna.htm.

[50] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull Math Biol*, 57:247–276, 1995.

[51] V. Pratt. Anatomy of the Pentium Bug. In P. D. Mosses, M. Nielsen, and M. I. Schwartzbach, editors, *TAPSOFT'95: Theory and Practice of Software Development*, number 915, pages 97–107. Springer Verlag, 1995.

[52] G. von Dassow, E. Meir, E. M. Munro, and G. M. Odell. The segment polarity network is a robust developmental module. *Nature*, 406:188–193, 2000.

[53] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In A. H. Taub, editor, *John von Neumann: Collected works, Volume V*, number 52 in Ann. of Math. Study, pages 329–378. Pergamon Press, New York, 1956.

[54] A. Wagner. *Robustness and evolvability in living systems*. Oxford University Press, 2007.

[55] M. Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 36(7):74–83, 1993.

APPENDIX

This appendix presents defining tables for Boolean functions associated to vertices of the Boolean network in Figure 2. The sign "-" stands for any value, 0 or 1. In each table, only the assignments of variables for which the function evaluates to 1 are shown; for all remaining assignments the function evaluates to 0.

LUG
1

CLF
1

UFO	UFO
1	1

LFY	SEP
1	1

AP3	FT
0	1

TFL1	AP2
0	1

LFY	EMF1
0	1

AP1	TFL1	FUL
0	0	1

WUS	AG	SEP	WUS
1	0	-	1
1	-	0	1

FT	LFY	AG	TFL1	AP1
1	-	0	-	1
-	1	0	-	1
-	-	0	0	1

AP1	EMF1	LFY	AP2	TFL1
0	1	0	-	1

FUL	AP1	EMF1	TFL1	LFY
1	-	-	0	1
-	1	-	0	1
-	-	0	-	1

AP1	LFY	AG	PI	SEP	AP3	PI
-	0	1	1	1	1	1
-	1	0	-	-	1	1
-	1	1	-	-	-	1
1	0	-	1	1	1	1

AP1	LFY	AG	PI	SEP	AP3	UFO	AP3
1	-	-	1	1	1	-	1
-	-	1	1	1	1	-	1
-	1	-	-	-	-	1	1

AP1	LFY	AP2	WUS	AG	LUG	CLF	TFL1	SEP	AG
-	-	0	-	-	-	-	0	-	1
-	1	-	-	1	-	-	-	1	1
-	1	-	-	-	-	0	-	-	1
-	1	-	-	-	0	-	-	-	1
0	1	-	-	-	-	-	-	-	1
-	1	-	1	-	-	-	-	-	1
-	1	0	-	-	-	-	-	-	1