Boolean networks with reliable dynamics

Tiago P. Peixoto* and Barbara Drossel†
Institut für Festkörperphysik, TU Darmstadt, Hochschulstrasse 6, 64289 Darmstadt, Germany
(Received 6 May 2009; published 6 November 2009)

We investigated the properties of Boolean networks that follow a given reliable trajectory in state space. A reliable trajectory is defined as a sequence of states, which is independent of the order in which the nodes are updated. We explored numerically the topology, the update functions, and the state space structure of these networks, which we constructed using a minimum number of links and the simplest update functions. We found that the clustering coefficient is larger than in random networks and that the probability distribution of three-node motifs is similar to that found in gene regulation networks. Among the update functions, only a subset of all possible functions occurs, and they can be classified according to their probability. More homogeneous functions occur more often, leading to a dominance of canalyzing functions. Finally, we studied the entire state space of the networks. We observed that with increasing systems size, fixed points become more dominant, moving the networks close to the frozen phase.

DOI: 10.1103/PhysRevE.80.056102 PACS number(s): 89.75.Da, 05.65.+b, 91.30.Dk, 91.30.Px

I. INTRODUCTION

Boolean networks (BNs) are used to model the dynamics of a wide variety of complex systems, ranging from neural networks [1] and social systems [2] to gene regulation networks [3]. BNs are composed of nodes with binary states coupled among each other. The state of each node evolves according to a function of the states from which it receives its inputs, similarly, to what is done when using cellular automata [4]; but in contrast to cellular automata, BNs have no regular lattice structure and not all nodes are assigned the same update function.

The simplest type of BNs are random BNs [5], where the connections and the update functions are assigned at random to the nodes. These random models have the advantage of being accessible to analytical calculations, thus, permitting a deep understanding of such systems [6]. Random BNs can display three types of dynamical behavior, none of which is very realistic: in the "frozen" phase, most or all nodes become fixed in a state which is independent of the initial conditions. In the "chaotic" phase, attractors of the dynamics are extremely long, and dynamics is very sensitive to perturbations. At the critical point between these two phases, attractor numbers are huge and depend strongly on the update scheme used [7,8].

In contrast to random BNs, real biological networks typically display a highly robust behavior. For instance, the main dynamical trajectory of the yeast cell-cycle network model derived by Li *et al.* [9] changes little when the nodes are updated in a different order, and the system returns quickly to this trajectory after a perturbation. In fact, whenever the functioning of a system depends on the correct execution of a given sequence of steps, the system must be robust with respect to the omnipresent effects of noise.

Motivated by this requirement, we focus in the present paper on the robustness of dynamical trajectories under fluctuations in the time at which the nodes are updated. We consider the extreme case, where we require the system to have a trajectory that is completely robust under a change in the update sequence. This means that at any time, all but one node would remain in their present state when they are updated. Recently, this question was also analyzed, albeit with a different approach, by Aracena *et al.* [10], where specific network constructions were investigated, which have this same property.

In contrast to the standard approach to BNs, where first the network structure (i.e., the topology and update functions) is defined and then the dynamics is investigated, we define first the dynamical trajectory and then construct networks that satisfy this trajectory, with the trajectory being robust under changes in the update sequence. A similar method has been used in [11]. In the next section, we will define the model and methods used. Then, we will discuss the properties of the networks constructed by these methods, considering the topology, the update functions, and the state space structure. Finally, we will outline directions for further investigations.

II. MODEL

A BN is defined as a directed network of N nodes representing Boolean variables $\boldsymbol{\sigma} \in \{1,0\}^N$, which are subject to a dynamical update rule,

$$\sigma_i(t+1) = f_i(\boldsymbol{\sigma}(t))u_i(t) + \sigma_i(t)[1-u_i(t)], \tag{1}$$

where f_i is the update function assigned to node i, which depends exclusively on the states of its inputs. The binary vector $\boldsymbol{u}(t)$ represents the *update schedule* and has components $u_i(t)=1$ if node i should update at time t, or $u_i(t)=0$ if it should retain the same value. The update functions f_i are conveniently indexed by the outputs of their truth table as follows. Given an arbitrary input ordering, each input value combination $\boldsymbol{\sigma}_j=\{\sigma_0,\sigma_1,\ldots,\sigma_{k-1}\}$ will have an associated index $c(\boldsymbol{\sigma})=\Sigma_i\sigma_i 2^i$, which uniquely identifies it. Any update function f can in turn be uniquely indexed by $f=\Sigma_j f(\boldsymbol{\sigma}_j)2^{c(\boldsymbol{\sigma}_j)}$, where $f(\boldsymbol{\sigma}_j)$ is the output of the indexed

*tiago@fkp.tu-darmstadt.de
†drossel@fkp.tu-darmstadt.de

 ©2009 The American Physical Society

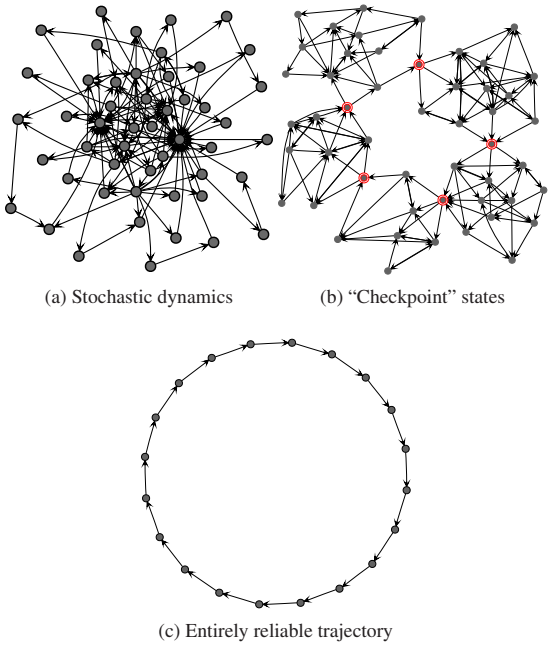


FIG. 1. (Color online) Illustration of levels of dynamical reliability. Each node on the graphs above is a state of the system, and the edges represent possible transitions between them.

function given the input value combination $\boldsymbol{\sigma}_j$.

The update schedule can be chosen in three different ways: (a) synchronous (parallel update), where $\boldsymbol{u}(t)=\mathbb{1}$, and all nodes are updated simultaneously every time step; (b) asynchronous and deterministic, where, for instance, $\boldsymbol{u}(t)=\{1-\Theta[(t+t_i^0)\text{mod } t_i]\}$, where t_i is the period with which vertex i is updated, t_i^0 is a local phase, and $\Theta(x)$ is the Heaviside step function; and finally (c) asynchronous and stochastic, where $u_j=1$ and $u_{i\neq j}=0$; in the fully stochastic case, j is a random value in the range $[1,N]$, chosen independently at each time step.

The choice of update schedule should take into account the fact that processes in biological (cellular) networks are subject to stochastic fluctuations, which can affect the timing of the different steps. In principle, a network could be organized such that the time interval between subsequent updates is so large that the update sequence is not affected by a small level of noise in the update times. In this case, an asynchronous deterministic updating scheme would be appropriate. However, more generally, the noise will also affect the sequence in which nodes are updated, suggesting an updating scheme that contains some degree of stochasticity.

In principle, networks can respond in different ways to stochasticity in the update sequence (see Fig. 1). (a) The system has no specific sequence of states, and it quickly looses memory of its past states. (b) The system has some degree of ordering in the sequence of states, with "checkpoint" states that occur in a given order and with certain groups of states occurring in between. (c) The system has entirely reliable dynamics, where the sequence of states is always the same on the attractor, no matter in which order the nodes are updated.

In this paper, we will focus on systems that have an attractor that has entirely reliable dynamics. Many cellular processes, such as the response to some external signal or the cell cycle, can only function correctly if the system goes through the required sequence of states in the correct order. Therefore, considering the idealization of fully reliable dynamics is biologically motivated. Furthermore, studying networks with entirely reliable dynamics is also of theoretical interest since it is an idealized situation on which one can build when studying more complicated cases. Entirely reliable dynamics can be implemented by enforcing that consecutive states of the attractor trajectory differ only in the value of one node. In other words, the Hamming distance between successor states is always 1. It is obvious that this is the only possible type of trajectory that can be entirely independent of the update schedule. If two subsequent states differed by the state of two or more nodes, then it would be possible to devise an update sequence, which would update one node but not the other, in contradiction to our assumption.

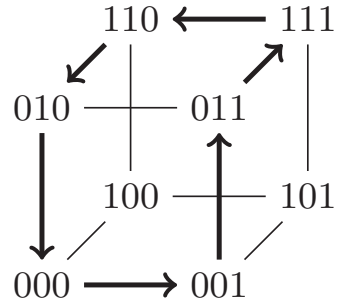


FIG. 2. Example of reliable trajectory of length 6 on a system of size $N=3$.

Entirely reliable attractors are represented in state space as simple loops. We denote the number of different states on the attractor by $L=\Sigma_i l_i$, where l_i is the number of times node i changed its state during a full period (since the trajectory is periodic, l_i must be equal to 0 or a multiple of 2). Furthermore, if the states of the system were represented by the corners of a N-dimensional Hamming hypercube, the trajectory should follow its edges (see Fig. 2). The shortest possible trajectory length, considering that no node remains at a constant value, is $L=2N$, with $l_i=2$ for all nodes. The longest possible trajectory length is $L=2^N$, where all states of the system are visited, and the trajectory corresponds to a Hamiltonian walk on the N-dimensional Hamming hypercube [18].

III. MINIMAL RELIABLE NETWORKS

A. Construction rule

The goal of this section is to construct BNs that have a given entirely reliable trajectory and to investigate their properties. A fully reliable trajectory has the property that the sequence of states is independent of the updating scheme, which means that under parallel update, only one node at a time changes its state. How networks that go through a given sequence of states can be constructed was demonstrated by Lau *et al.* [11], who investigated all possible networks, which exhibit the main trajectory of the yeast cell-cycle regulatory network. Thus, we first define the dynamics, from

which we obtain the topology and functions, opposite to what is usually done in the literature on Boolean networks.

In fact, there exist many networks that display a given trajectory. Even when the full state space structure is specified, which defines the successor state of each of its 2^N possible states, it is possible to construct a network that has this state space structure. This can be done by constructing a fully connected graph with $k=N$ and by assigning to each node the function that has the required output for each of the 2^N input states. In the end, inputs that never affect the output can be removed. If there are different sets of inputs that can be simultaneously removed, different networks are obtained.

When not the entire state space structure but only one reliable trajectory is specified, there exist consequently many networks with different topology and functions, which have this trajectory and may differ in the rest of their state space. We will restrict ourselves to *minimal* networks, i.e., networks with the smallest possible number of inputs for each node and the simplest possible functions, which have the maximum possible number of identical entries in the truth table. This minimality condition is motivated by the putative cost associated with more connections or more complicated functions, which would decrease the fitness of an organism. This is in contrast to what was done in [11], where all possible networks were considered, which is only feasible on very small systems.

Such minimal networks can be constructed by a straightforward algorithm because the inputs and the function required for each node can be determined independently from those of all the other nodes. The inputs for a given node must include all predecessor nodes, which change their state 1 time step before the given node changes its state. Additional inputs are required if the given node assumes, during the course of the trajectory, different binary states for the same configuration of the predecessor nodes. The choice of these "excess" inputs is usually not unique and may include self-inputs. We perform this choice at random but only from the possibilities which minimize the number of inputs to each node. If not all possible configurations of the states of the input nodes occur during the course of the trajectory, the update function of the given node is not unique. We first assign those truth table entries of the update function that are specified by the trajectory. Then, we assign to all remaining entries the same output value, and we choose the majority of output values assigned so far. (If there is no majority, we choose either value with probability 1/2.)

The algorithm used for choosing the minimal set of inputs proceeds as follows. To each node, we first assign all predecessor nodes as inputs. Then, if needed, we choose excess inputs. We first set the number of excess inputs to $k'=1$, and we test in a random order the $\binom{N}{k'}$ possible node combinations until we find a node set which, together with the predecessors, is a valid input set. If no valid combination is found, we increase k' by 1 and repeat the search. Once a valid combination is found, the corresponding truth table is completed by applying the minimality condition to its unspecified entries. The run time of this algorithm increases as $O(lN^{\max(\max(k'),1)})$, where l is the average number of flips per node, and $\max(k')$ is the maximum value of k' for all nodes.

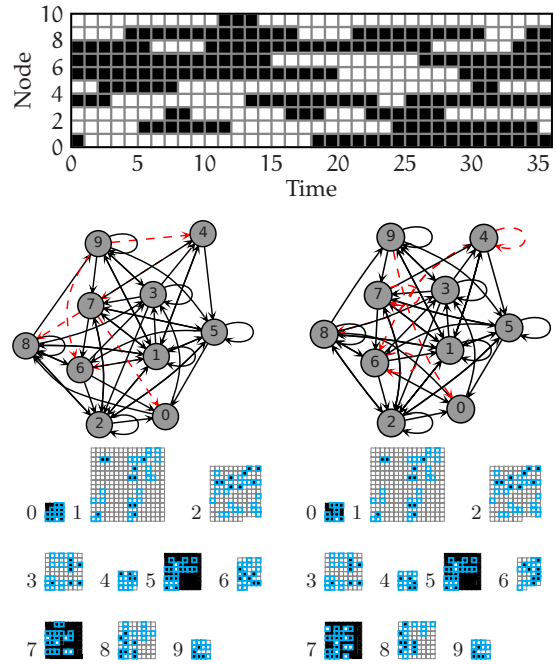


FIG. 3. (Color online) Example of a random reliable trajectory for $N=10$ and $l=4$, and two possible minimal networks. The edges with dashed (red) lines represent the inputs that are different between the two networks. Below each network are the outputs of the truth table of each node ordered from top to bottom and left to right, according to their input combination indices. Outputs marked in gray (cyan) correspond to input combinations present in the given trajectory.

We have observed that the run times are feasible for networks of size up to $N=400$ and $l=12$ [19].

An example for a reliable trajectory and two possible networks with their functions obtained with the above algorithm is given in Fig. 3.

We choose the reliable trajectory at random, without taking into consideration possible particular features of biological networks, such as different temporal activation patterns of the different nodes, which reflect the function that the network must fulfill. Instead, we will consider a null model, where the values of the nodes change randomly. The only restriction, which is imposed, is that the trajectory is reliable. The only two parameters of this trajectory are the number of nodes N and the average number of flips per node l. We generate a random ensemble of reliable trajectories in the following way. First, we determine how often each node shall be flipped. To this purpose, for each node i a random number λ_i is chosen from a Poisson distribution with mean $(l-2)/2$, implying that node i shall be flipped $l_i \equiv 2\lambda_i+2$ times. The average number of flips of each node is thus identical to l, and each node is flipped at least twice. The length of the trajectory is then $L=2N+2\Sigma_i\lambda_i=\Sigma_i l_i$. Then, we arrange these flips in a randomly chosen order. If the resulting trajectory contains the same network state twice, it is discarded, and a new sequence of flips is chosen.

B. Topological characteristics

We first present results for the topological characteristics of the obtained networks. We evaluate the degree distribution

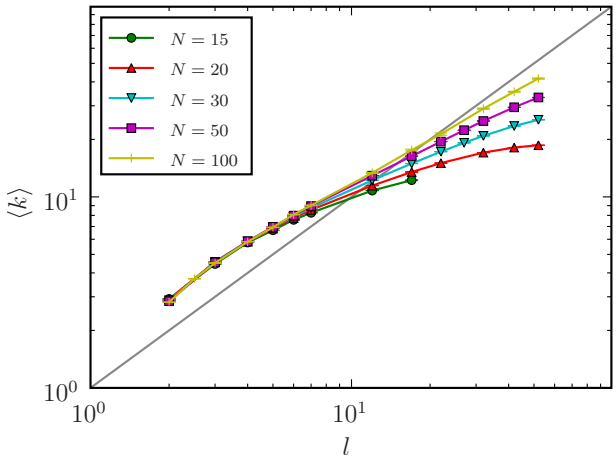


FIG. 4. (Color online) Average degree $\langle k \rangle$ as a function of l for networks of different size N. The straight line is the function $\langle k \rangle = l$.

and the local correlations. The degree distribution is of course strongly dependent on l. Local correlations can arise when two nodes that are influenced by the same nodes are more likely to influence each other.

Unless stated otherwise, we averaged the results from several independent realizations of the minimal trajectories and minimal networks, for different N and l. The number of realizations for small N, up to 20, were at least 2000. For intermediary values of N, up to 100, it varied from 50 to 300, depending on l. For the larger networks $N > 100$, it ranged from 200 to 6 networks for $l < 12$, and one realization for $N = 400$ and $l = 12$.

1. Degree distribution

The number of inputs of a node is at least as large as the number of its predecessors. Whenever the state of the node cannot be written as a function of the predecessors alone, excess inputs must be chosen, as already mentioned before. The number of different predecessors n_p per node approaches, for large N, on average l, since it becomes unlikely for large N that the same node is chosen twice as predecessor. The typical truth table size grows therefore with l as 2^l. Since the number of different predecessor states grows only quadratically as $n_p l \sim l^2$, one can expect the number of excess inputs to be small, and the number of inputs per node should be

$$\langle k \rangle \simeq l, \tag{2}$$

for sufficiently large N. This is confirmed by our numerical investigations, as is shown in Fig. 4.

The degree distribution mirrors the distribution of the number of predecessors. Since all nodes flip on average the same number of times, the distribution is expected to follow a Poisson distribution for large enough l. This is indeed the case, as Fig. 5 shows. For small l, however, the distributions are more narrow because we imposed the condition that each node flips at least twice, leaving little freedom for additional predecessors when l is close to 2.

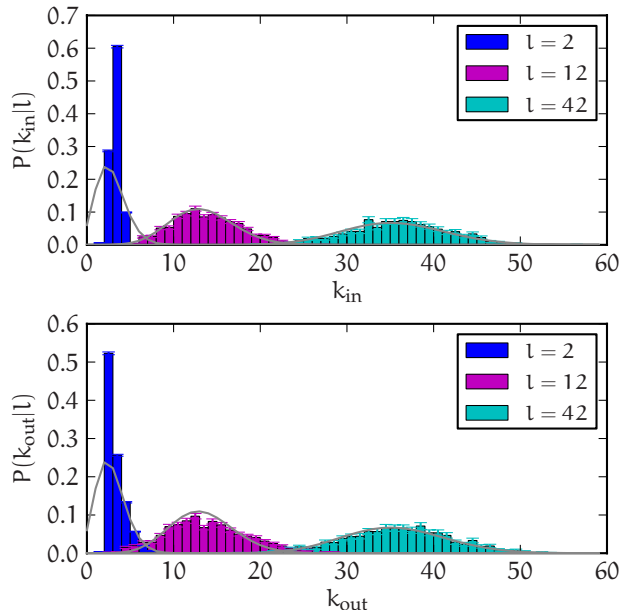


FIG. 5. (Color online) In-degree and out-degree distributions of minimal networks for different values of l, for $N = 100$. The solid lines correspond to Poisson distributions with the same average.

2. Local correlations

We obtained information about the local topology of the minimal networks by evaluating the probability that the neighbors of a given node are connected to each other. This probability is the so-called *clustering coefficient* $\langle c \rangle$ [12]. Random uncorrelated networks show the absence of clustering only in the limit $N \to \infty$. Thus, for finite N, it is necessary to compare the obtained value with a random network of equal size and with equal degree distribution. In order to do this, we calculated the clustering coefficient $\langle c_s \rangle$ on *shuffled* networks, where the links were rewired randomly, preserving the in and out degree of each node. We then calculated the ratio $\langle c \rangle / \langle c_s \rangle$, for networks of different size and average flip number l. If the ratio approaches 1, the network does not exhibit any special clustering. The results for several values of N and l are shown in Fig. 6.

The most evident feature of Fig. 6 is that clustering is stronger for smaller l, i.e., for sparse networks. For larger l (and hence larger $\langle k \rangle$), the average distance between nodes decreases, and the shuffled and original networks have a similar degree of clustering. This difference between networks with smaller and larger average degree becomes more pronounced when the size of the networks N is increased. From the data in Fig. 6, it appears that the ratio $\langle c \rangle / \langle c_s \rangle$ increases slowly with N. We will argue in the following that this ratio will reach a finite asymptotic value in the limit $N \to \infty$.

The finding that the clustering coefficients are larger than for random networks can be explained by considering the above-mentioned excess inputs that are required when the function assigned to a node cannot be based on its predecessor nodes alone. Let us consider two consecutive flips of a node j on a given trajectory. These flips are preceded by flips of the predecessor nodes, which we call v and w. The aver-

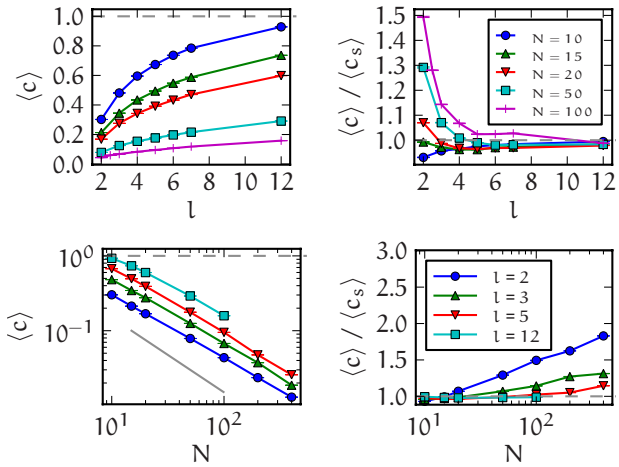


FIG. 6. (Color online) Clustering ratio $\langle c\rangle/\langle c_s\rangle$ as a function of the average number of flips per node l, for different network sizes N. The gray straight line corresponds to a decay of the type $1/N$.

age time between the two considered flips of node j is $\sim L/l=N$, implying that there is a considerable probability that node v flips again before the second flip of node j, giving the sequence

$$vj\cdots v\cdots wj.$$

The update function assigned to node j needs an excess input if neither node w nor any other predecessor of node j (which can exist only for $l>2$) flips between the first flip of j and the second flip of v. The simplest choice of this excess input is the node j itself. Indeed, self-inputs occur more often than in the shuffled networks, as is shown in Fig. 7. Since the number of different possible excess inputs is proportional to N, we expect that the fraction n_l of nodes with self-inputs decreases as $n_l\sim 1/N$ for large N but remains larger than that of shuffled networks by a constant multiplicative factor.

The excess input cannot be a self-input if node w flips also in the same interval, giving the sequence

$$vj\cdots v\cdots w\cdots wj.$$

In this case, an excess input u must be chosen among those nodes that flip between the two consecutive flips of node w,

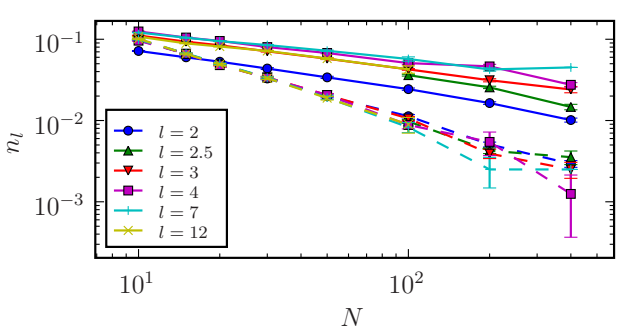


FIG. 7. (Color online) Fraction n_l of nodes with self-input, as a function of N, for different values of l. The dashed curves are obtained for shuffled networks, with the same degree sequence. The inset shows the ratio n_l/n_l^s, where n_l^s is the self-input ratio for the shuffled networks.

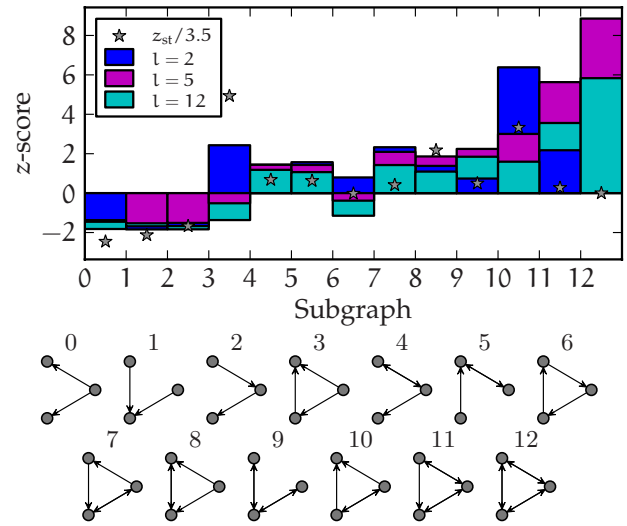


FIG. 8. (Color online) The z score of the different three-node subgraphs of minimal reliable networks, for different values of l and $N=100$. The profile z_{st} corresponds to the signal-transduction interaction network in mammalian cells [15].

if none of the other predecessors of j flips in this interval, giving the sequence

$$vj\cdots v\cdots w\cdots u\cdots wj.$$

Now, the average distance between the flips of node w and node u is smaller than that between two randomly chosen nodes since w is required to flip in the indicated interval. Therefore, the probability that w is an input to u or vice versa is larger than random, and it scales as $1/N$ in the limit $N\to\infty$. Since w and u are inputs to j, it follows that the clustering coefficient is larger than the random value $\langle c_s\rangle$.

From this consideration, it follows that the ratio $\langle c\rangle/\langle c_s\rangle$ approaches a constant value in the limit $N\to\infty$. Furthermore, it follows that this ratio is larger for smaller l since it is less likely that there exist additional inputs to j that flip in the required interval and make excess inputs unnecessary. The slight increase seen in Fig. 6 can probably be attributed to a finite-size effect.

In order to determine which three-node subgraphs contribute to the increased clustering, we evaluated their z score, which indicates to what extent the frequency of each subgraph is different compared to the random case. The z score is defined as

$$z_i=\frac{\langle N_i\rangle-\langle N_i^s\rangle}{\sqrt{\langle (N_i^s)^2\rangle-\langle N_i^s\rangle^2}}, \tag{3}$$

where N_i is the number of occurrences of subgraph i, and N_i^s is the number of occurrences of the same subgraph on a shuffled network with the same degree sequence. Figure 8 shows the different possible subgraphs and their z score. Subgraphs with more links have a higher z score and are therefore *network motifs*. Sparser subgraphs, where there is no link between two of the nodes, are rarer than at random, as predicted by the clustering coefficient. The abundance of denser motifs increases with l, as the network itself becomes

more dense, but the overall trend of the z score is the same. One peculiar feature is the absence of simple loops (subgraph 6), also known as feedback loops [13]. As was described above, the clustering is mostly due to the correlations between the inputs of a given node or a reciprocal correlation between a node and one of its inputs. A simple loop is not caused by this type of correlation. Furthermore, it was shown by Klemm *et al.* [14] in a study of the reliability of small Boolean networks that feedback loops are harmful to reliable dynamics. These authors obtained a z-score profile very similar to Fig. 8 (see Fig. 4 of [14]). They also showed that this profile is qualitatively similar to real biological networks studied in [15]. A direct comparison is shown in Fig. 8, with the motif profile of the signal-transduction interaction network in mammalian cells [15]. We note that while mere simple loops are suppressed, they occur as a subgraph of the subgraphs 9, 11, and 12, which have a higher z score. But the higher z score of these subgraphs is due to the larger number of total or reciprocal links caused by the above-mentioned correlations. For a given number of total and reciprocal links, subgraphs with a feedback loop have a lower z score than subgraphs without a feedback loop.

In Fig. 8, we did not keep track of the self-inputs, for simplicity. When self-loops are included in the subgraphs, their number increases from 13 to 86, which makes the analysis and presentation more elaborate. We performed this analysis and found that a subgraph with a specific number of self-loops has a larger z score than its counterpart with less or no self-loops. The z-score pattern of Fig. 8, on the other hand, is repeated for subgraphs which share the same number of self-loops, which shows that motif occurrence and self-regulation are largely independent.

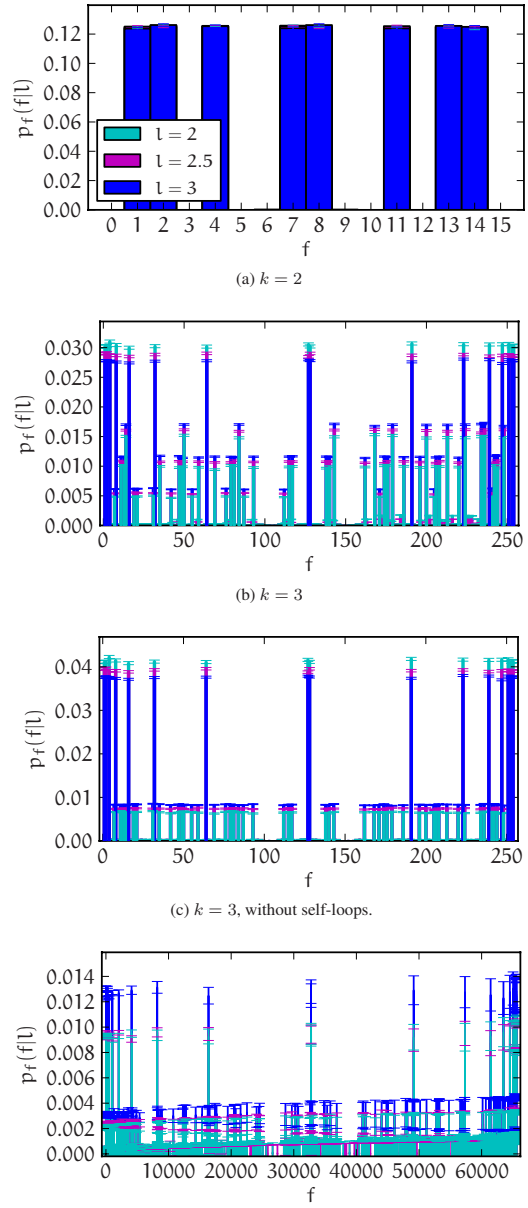
C. Properties of update functions

We evaluated the frequency of the different types of update functions in minimal networks, for different values of l (see Fig. 9). Unless otherwise stated, the results were obtained from 10^4 independent realizations of networks with $N=20$. We compared the results with those obtained for larger values of N, with no discernible difference other than the reduced statistical quality. Functions with different numbers of inputs were evaluated separately.

The functions seem to be distributed according to different classes, where functions of the same type occur with the same probability, while some do not occur at all. In order to understand this distribution, it is necessary to describe in detail what conditions need to be met by the functions, according to the imposed dynamics and construction rules.

The subsystem composed only of the inputs of a given node follows a certain "local trajectory" (i.e., sequence of states), which determines, together the minimality condition described in Sec. III, the update function of the considered node. The probabilities of the different possible trajectories depend on the way the global trajectory is specified and on the rules for choosing excess inputs. The restrictions imposed on the local trajectories of the inputs are as follows:

(1) The local trajectory of the inputs must correspond to a periodic walk on the k-dimensional hypercube representing



(a) $k=2$

(b) $k=3$

(c) $k=3$, without self-loops.

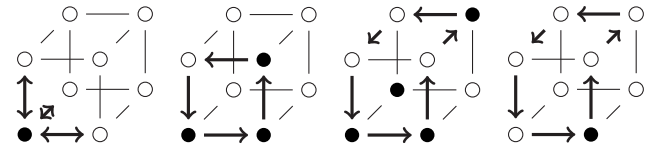
(d) $k=4$, without self-loops.

FIG. 9. (Color online) Distribution of the different update functions, for different numbers of inputs k and for different flip numbers l, for networks of size $N=20$.

their states since the Hamming distance at each step must be 1. We note that in this subsystem, the same input state is allowed to repeat within a period (only the global state cannot). The vertices of the hypercube can be annotated with the output value of the function at the corresponding input state (see Fig. 10, for examples).

(2) For large N, the trajectories of any two different nodes will be approximately random and uncorrelated. The only restriction is that every face of the hypercube will be visited exactly l_v times, where v is the index of the input node that has a fixed state on this face. On average, we have $\langle l_v \rangle = l$.

(3) The output values of the function can be distributed on the vertices of the hypercube that are visited during the walk in any possible way, with the restriction that the output value



(a) Valid trajectory (b) Invalid (restriction 2) (c) Invalid (restriction 4) (d) Invalid (restriction 5)

FIG. 10. Example of input and output trajectories on the k hypercube representing the states of the inputs, for functions with $k=3$. Allowed transitions are represented by arrows. The color on each vertex represents the output value. (a) represents one type of valid trajectory. (b)–(d) represent invalid trajectories, according to the indicated restriction: (b) not all sides of the cube are visited; (c) the function is not minimal; and (d) the function can be reduced to $k=2$, since flipping the third input (i.e., going to the rear face and back) does not change the state of the node.

must change l_j times along the walk, where j is the index of the considered node. An exception are functions with self-inputs: the vertices on the hypercube face corresponding to the self-input must all have the same output value.

(4) The output values at the vertices of the hypercube, which are not visited by the walk, must be equal to the majority of the output values on the walk (this is the minimality condition defined in Sec. III).

(5) Functions that can be reduced to a function with smaller k cannot occur due to the minimality condition, and the corresponding trajectory can be confined to a hypercube of smaller dimension.

Figure 10 shows examples of trajectories that are allowed or not allowed for the case $k=3$.

The listed restrictions result in the observed distribution of update functions. We will describe in detail all the possibilities for $k=2$ and discuss in a more general and approximate manner the functions with $k>2$.

1. Functions with k=2

Figure 9 shows that only eight of the 16 possible functions occur, and all of them with equal probability. They are all *canalyzing functions*, with three entries 1 (or 0) in the truth table and one entry 0 (or 1). The hypercube representation of all functions is shown in Fig. 11. The functions that are not possible are obviously the constant functions (first row of Fig. 11, from left to right) and the functions which are insensitive to one of their inputs, due to restriction 4 (second and third rows). The other functions, which do not occur, are the reversible functions, which change the output at every change in an input (fourth row). Those functions, however,

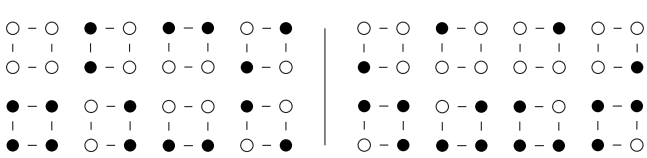
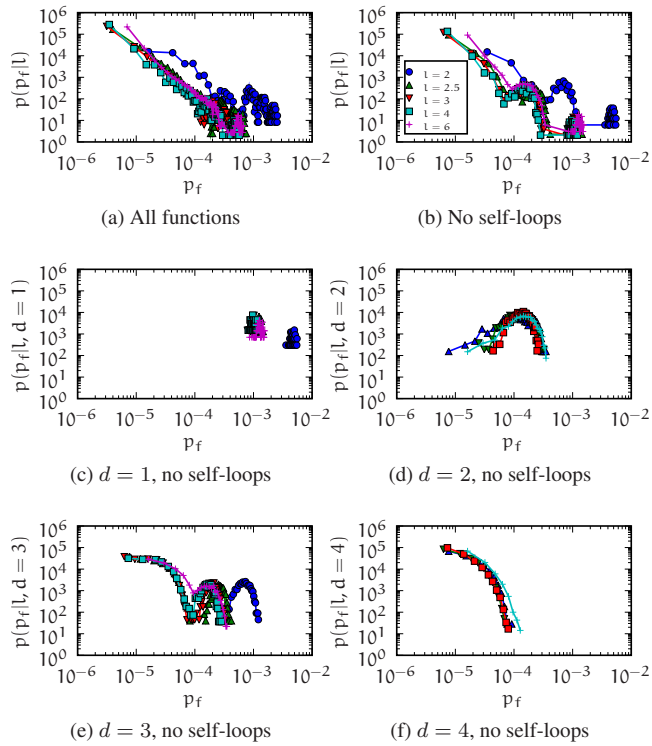


FIG. 11. Representation of all 16 functions with $k=2$ on the 2–hypercube. On the left are the functions, which do not (or rarely) occur in the minimal networks, and on the right are the canalyzing functions, which occur with equal probability.



(a) All functions (b) No self-loops (c) $d=1$, no self-loops (d) $d=2$, no self-loops (e) $d=3$, no self-loops (f) $d=4$, no self-loops

FIG. 12. (Color online) Distribution of function weights p_f subdivided according to the value of the truth table homogeneity d, for different values of the average flip number l, for fixed values $k=5$, and $N=20$.

are not entirely impossible. It is possible to construct a trajectory that meets all the listed requirements, with the specification that the output flips as often as all inputs together (restrictions 2 and 3). Such trajectories follow the pattern:

$$vj \cdots wj \cdots vj \cdots wj,$$

where v and w are the inputs of j. This pattern is impossible for $l=2$ but can occur for larger l, albeit with a small probability, since $k=2$ functions are less likely for larger l; furthermore, the probability that a node has two predecessors, which occur twice, decreases with N as $\sim 1/N^2$.

2. Functions with k>2

Functions with $k>2$ seem to fall into different classes, which occur with different probabilities. This can be seen by plotting the distribution of the probabilities p_f of the different functions, as shown in Fig. 12(a) for $k=5$. The different classes seem to correspond to different function homogeneity values defined as the number of minority output values in the truth table d. This can be verified by selecting only those functions with a given value of d and plotting their distribution of probabilities, as shown in Figs. 12(c)–12(f). The most frequent class comprises the functions with only one entry in the truth table deviating from the others ($f=2^i$ and $f=2^k-2^i$), with $d=1$ [see Fig. 12(c)]. Those are *canalyzing* functions, where all inputs are canalyzing inputs. Functions with the same homogeneity fall into subclasses, which have different probabilities. Those functions are often negated functions ($f'=2^k-f$) of one another, and this is due to the existence of

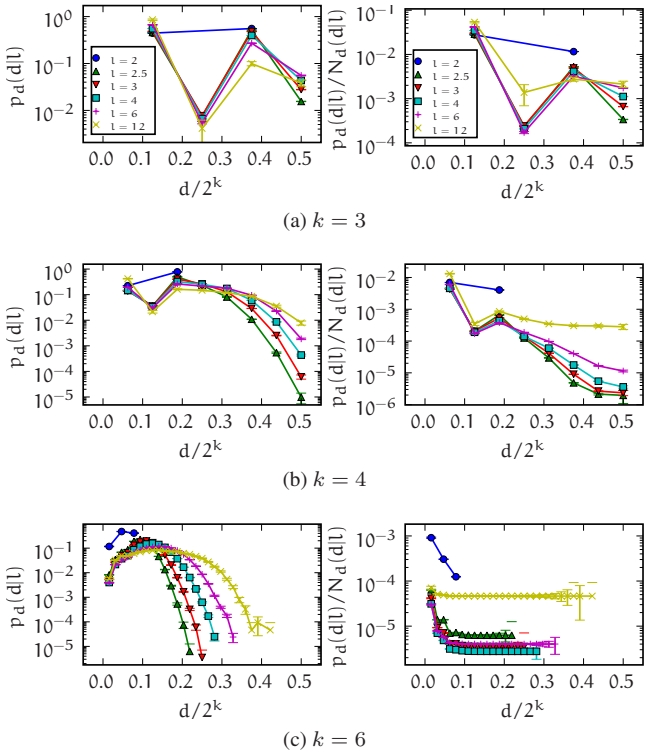


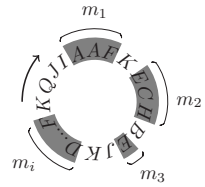
FIG. 13. (Color online) Distribution of functions with different values of the truth table homogeneity d, for different average flip number l, and $N=20$.

self-inputs. Self-regulated functions are not equivalent functionally when they are negated (the input corresponding its own output must be negated as well), despite sharing the same homogeneity. The $0 \leftrightarrow 1$ symmetry, however, is always preserved. When self-loops are ignored, the distribution becomes symmetric with respect to negation of the output [see Fig. 9(c)], and the homogeneity classification becomes the predominant criterion to distinguish between the classes [compare Figs. 12(a) and 12(b)]. But even in the absence of self-loops, the probability classes are not uniquely defined by the homogeneity, and there are overlaps between the different classes, as Figs. 12(d)–12(f) show. Nevertheless, there is a general tendency that functions with larger d are less likely.

Figure 13 shows the probability of finding a function with a given value of d. Since the number of different functions in a given class increases rapidly with d for small d, the maximum of this distribution is shifted to values of d larger than 1. If this distribution is corrected by the number N_d of different functions found with the same value of d, an overall decreasing function of d is obtained, as shown in the graphs in the left column of Fig. 13).

The observed difference in probability due to different homogeneity can be explained as follows. We consider a node with k inputs. We denote by $M=\Sigma_i m_i \in [l/2, L-l/2]$ the total time during which the node is in the state that it assumes less often. The sum is taken over all intervals during which the node has this state.

If we denote the different possible (combined) states of the input nodes by letters, we can represent the sequence of states through which the considered node and its input states go by the following picture:



the shaded areas correspond to the output value 1. A state of the input nodes that appears inside the shaded (clear) area must appear again inside the shaded (clear) area each time it is repeated. If we consider only the above scenario and essentially ignore that the trajectories must follow the edges of a k hypercube, we can show that functions with smaller values of d should occur more often.

Our approximations rely on the fact that, for $N \to \infty$ and $l \gg 1$ (and hence $L \to \infty$), the shaded areas will be more numerous and will be further apart in time and less correlated. In this limit, the input state number i occurs, say, n_i times. The probability that each of the input states occurs only in one type of area is given approximately by

$$\prod_i \left[\left(\frac{M}{L} \right)^{n_i} + \left(\frac{L-M}{L} \right)^{n_i} \right]. \tag{4}$$

The maximum of this function is attained at $M=l/2$ (or $M=L-l/2$, which is excluded since we chose M such that it counts the minority part), which is the minimal possible value. The value of d is bounded by M but can be smaller since the same input state can repeat. We can in fact see that the case, where the same state repeats at all M times, is more probable, by considering all the possible permutations of the state sequence, for a given value of d,

$$\left[\prod_{i \le d} \binom{M - \sum_{j<i} n_j}{n_i} \right] \left[\prod_{i>d} \binom{L - M - \sum_{d<j<i} n_j}{n_i} \right] \tag{5}$$

and observing that it has a maximum at $d=1$, since $M \ll L$. (This means that there are M shaded areas of size 1 each.) It follows that with increasing l, the weight of update functions with $d=1$ will become much larger than that of every other update function, as is evident from Figs. 9 and 13. The dominance of $d=1$ functions can already be seen for small values of l, although it is less pronounced.

D. State space structure

Finally, we investigated the state space of the constructed networks. We considered the system under a stochastic update scheme since this scheme underlies the study presented in this paper. In this case, we define an attractor as a recurrent set of states in the state space, with the property that there are no transitions that escape this set (i.e., a strongly connected component in the state space graph that has no outgoing connections). The number of states in this set is called the size of the attractor.

We evaluated the probability that an attractor of a given size occurs in a minimal network and its average basin size. For small networks (up to $N=12$), these results were obtained by exact enumeration of the state space. For larger N, the state space was sampled, taking care that the same attrac-

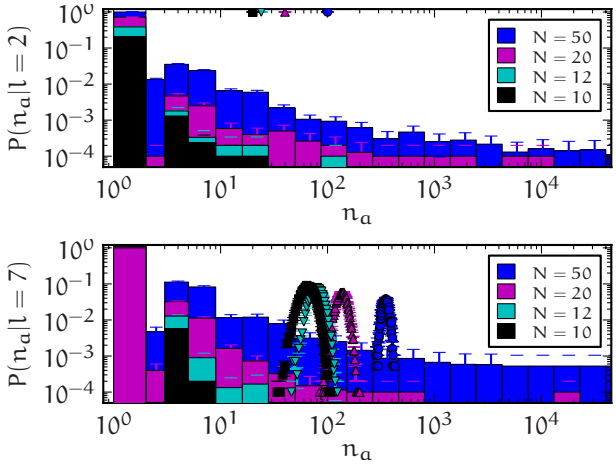


FIG. 14. (Color online) Probability of attractor sizes n_a, for $l=2$ and $l=7$. Attractors corresponding to the given trajectories are plotted separately with symbols. For each value of N and l, 10^4 different networks were analyzed. In the case of attractor sampling, 100 different random initial conditions per network were used.

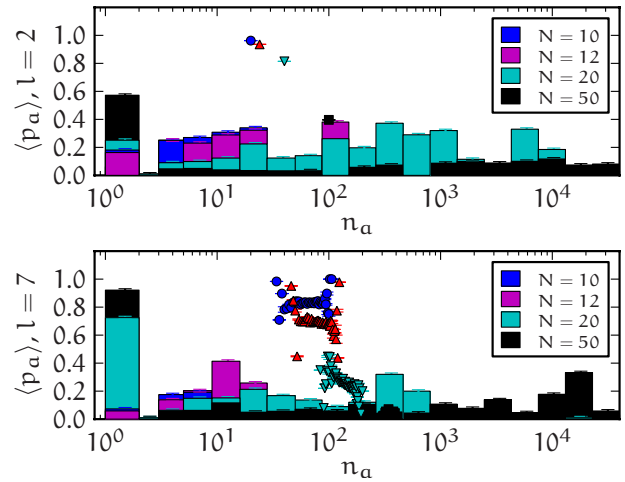


FIG. 15. (Color online) Average attractor probabilities (basin size) $\langle p_a \rangle$, for $l=2$ and $l=7$. Attractors corresponding to the given trajectories are plotted separately with symbols. For each value of N and l, 10^4 different networks were analyzed, and 100 different random initial conditions per network were used.

tor was not counted twice. This method, however, leads to a bias since attractors with smaller basins are less likely to be counted, and the extent of this bias depends on the size of the network. Nevertheless, this bias is not relevant for our point of interest, which is on the occurrence of various attractors, but not on their precise statistics. Figure 14 shows that there exist almost always fixed points and that there are often attractors, which are much larger than the imposed reliable trajectory (we considered attractors of up to $n_a=10^5$ states). Note that the probabilities in Fig. 14 do not sum to 1 but to the average number of different attractor sizes occurring in a network.

The basin of attraction was measured as the probability of reaching an attractor, starting from a random configuration, averaged over several networks. We note that similarly to the situation above, the probabilities need not sum to 1. Figure 15 shows that the omnipresent fixed point has a large basin of attraction. Larger attractors occur with smaller probabilities. The weight of the fixed point compared to the weight of the imposed reliable trajectory increases with increasing N. This can be explained by the entries in the truth table, which are not uniquely determined by the reliable trajectory: While number of entries fixed throughout the trajectory grows linearly with l, the number of remaining entries (as well as their contribution to the state space) grows exponentially. In this increasingly large region of the state space, the functions behave as constant functions.

Attractors, which are larger than the given trajectories are due to a portion of network, begin frozen in the value they have at the fixed point, while other nodes remain frustrated, and their states change stochastically, visiting a larger portion of the state space, without entering the fixed point or the reliable trajectory.

For comparison, we briefly looked at the attractor sizes obtained using a synchronous updating scheme. Not surprisingly, the attractors become much shorter in this case, with attractors larger than the given trajectory having only a small probability (not shown).

IV. CONCLUSION

We have constructed minimal Boolean networks, which follow a given reliable trajectory in state space. The trajectories considered have the necessary feature that only one node can change its value at any moment in time, which guarantees that the sequence of states is independent of the order in which nodes are updated. Otherwise, the nodes change their states at randomly assigned times in the given trajectory, thus, constituting a null model for reliable dynamics. The minimality condition imposed on the networks was that it contains the smallest possible set of inputs for each node that allows for the given trajectory. Additionally, the truth table entries that are not fixed by the trajectory were set to the majority value imposed by the trajectory. We then investigated the topology, the update functions, and the state space of those networks.

The network structure, as manifest in the degree distribution, does not deviate significantly from a random topology. However, the network exhibits larger clustering than a random network and exhibits a characteristic motif profile, which resembles both real networks of gene regulation and the pattern of dynamically reliable motifs found in [14]. The existence of clustering and motifs was explained by considering the excess inputs that are required to avoid contradictions in the truth table and how they must be correlated among each other.

The update functions of the nodes show a characteristic distribution, where only a subset of the possible functions occur, and these are divided into distinct classes, which occur with different probabilities. The main factor discerning the different classes is their homogeneity characterized by the number of entries of the minority bit in the truth table. Functions with homogeneity 1 occur with increased probability and become dominant functions in the limit of large trajectories $l\to\infty$, for fixed k. Functions with more minority entries occur with a smaller probability, and this probability

decreases as the number of minority entries increase. We presented an analytical justification for this finding, considering how the local trajectory of the input states of a given function must behave, in the limit $l \gg 1$.

Finally, we investigated the state space of the constructed networks, considering the possible attractors it can have, in addition to the given reliable trajectory. To this aim, we used a stochastic update scheme. We observed that the dynamics of the network almost always exhibits a fixed point and often attractors which can be much larger than the given trajectory. The basin size of the fixed point is very large and dominates the basin size of the given trajectory in the limit of large system size. This is a consequence of the minimality condition imposed on the network: the region of state space dictated by the imposed trajectory increases only linearly with system size, while the entire state space grows exponentially. Outside the state space region fixed by the reliable trajectory, the constructed functions behave as constant functions, which drive the system nearer to the frozen phase.

In this work, we have used a null model for reliable trajectories, where the nodes change their values at random times. Real gene regulatory networks deviate significantly from this since they must agree with the cell cycle or the pathway taken during embryonic development. Certain proteins need to be always present in the cell, while others are produced only under specific conditions. The degree distribution and the update functions must reflect this behavior. However, some of the features found for the null model presented in this work should also be present in more realistic systems. The existence of clustering and the motif profile found, for instance, do not depend strongly on the specific temporal patterns of the nodes but are imposed by the reliability condition. Similarly, the dominance of strongly canalyzing functions is a consequence of the reliability condition and should be relatively robust to the introduction of temporal correlations. Nevertheless, biochemistry makes some canalyzing functions more likely than others.

An important feature of biological networks that is not reflected in the null model presented in this paper is the robustness with respect to perturbations of a node. Such a robustness can only be obtained when more than the minimum possible number of inputs is assigned to a node. Indeed, it has been shown in [16] that more redundancy allows for more robustness. Similarly, requiring that the reliable trajectory has the largest basin of attraction or that other attractors of the system are also reliable trajectories may increase the number of links in the network.

Finally, the requirement that trajectories are fully reliable is an idealization, which goes beyond what is necessary for gene regulatory networks. Real networks have checkpoint states, but between these states, the precise sequence of events is not always important. On the other hand, full reliability may be necessary for certain subsystems of the gene network, where a strict sequence of local states is required. The minimal reliable networks discussed in this paper should be compared more realistically to such reliable modules.

ACKNOWLEDGMENTS

We thank Ron Milo for providing the signal-transduction network data. We acknowledge the support of this work by the Humboldt Foundation and by the DFG under Contract No. Dr300/5-1.

[1] M. Rosen-Zvi, A. Engel, and I. Kanter, Phys. Rev. Lett. **87**, 078101 (2001).
[2] A. A. Moreira, A. Mathur, D. Diermeier, and L. A. N. Amaral, Proc. Natl. Acad. Sci. U.S.A. **101**, 12085 (2004).
[3] M. C. Lagomarsino, P. Jona, and B. Bassetti, Phys. Rev. Lett. **95**, 158701 (2005).
[4] S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, IL, 2002).
[5] S. A. Kauffman, J. Theor. Biol. **22**, 437 (1969).
[6] B. Drossel, in *Reviews of Nonlinear Dynamics and Complexity*, edited by H. G. Schuster (Wiley, New York, 2008), Vol. 1.
[7] F. Greil and B. Drossel, Phys. Rev. Lett. **95**, 048701 (2005).
[8] K. Klemm and S. Bornholdt, Phys. Rev. E **72**, 055101(R) (2005).
[9] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, Proc. Natl. Acad. Sci. U.S.A. **101**, 4781 (2004).
[10] J. Aracena, E. Goles, A. Moreira, and L. Salinas, BioSystems **97**, 1 (2009).
[11] K. Y. Lau, S. Ganguli, and C. Tang, Phys. Rev. E **75**, 051907 (2007).
[12] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[13] U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman & Hall, London/CRC, London, 2007).
[14] K. Klemm and S. Bornholdt, Proc. Natl. Acad. Sci. U.S.A. **102**, 18414 (2005).
[15] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, Science **303**, 1538 (2004).
[16] C. Gershenson, S. A. Kauffman, and I. Shmulevich, in *Artificial Life X, Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, edited by L. S. Yaeger, M. A. Bedau, D. Floreano, R. L. Goldstone, and A. Vespignani (MIT Press, Cambridge, MA, 2006), pp. 35–42.
[17] D. E. Knuth, *The Art of Computer Programming*, Fascicle 3: Generating All Combinations and Partitions Vol. 4 (Addison-Wesley, Reading, MA, 2005).
[18] Unlike in some types of graphs, those trajectories are always possible on hypercubes and are known as Grey codes in computer science [17].
[19] We note that an alternative procedure that starts with an input set that contains all nodes and then randomly removes inputs, until a removal is no longer possible, is for larger k much faster than the procedure used by us. However, it will in general not produce a minimal network. As an example, consider the case where only one minimal set of k inputs is possible for a given node. In this case, if one of these inputs is removed early in the iteration, the input set obtained at the end will have a size larger than the minimal k.