Attractor Stability in Finite Asynchronous Biological System Models

Henning S. Mortveit\(^1\) · Ryan D. Pederson\(^2\)

Received: 15 November 2016 / Accepted: 27 December 2018 / Published online: 17 January 2019
© Society for Mathematical Biology 2019

Abstract

We present mathematical techniques for exhaustive studies of long-term dynamics of asynchronous biological system models. Specifically, we extend the notion of \(\kappa\)-equivalence developed for graph dynamical systems to support systematic analysis of all possible attractor configurations that can be generated when varying the asynchronous update order (Macauley and Mortveit in Nonlinearity 22(2):421, 2009). We extend earlier work by Veliz-Cuba and Stigler (J Comput Biol 18(6):783–794, 2011), Goles et al. (Bull Math Biol 75(6):939–966, 2013), and others by comparing long-term dynamics up to topological conjugation: rather than comparing the exact states and their transitions on attractors, we only compare the attractor structures. In general, obtaining this information is computationally intractable. Here, we adapt and apply combinatorial theory for dynamical systems from Macauley and Mortveit (Proc Am Math Soc 136(12):4157–4165, 2008. https://doi.org/10.1090/S0002-9939-09-09884-0; 2009; Electron J Comb 18:197, 2011a; Discret Contin Dyn Syst 4(6):1533–1541, 2011b. https://doi.org/10.3934/dcds.2011.4.1533; Theor Comput Sci 504:26–37, 2013. https://doi.org/10.1016/j.tcs.2012.09.015; in: Isokawa T, Imai K, Matsui N, Peper F, Umeo H (eds) Cellular automata and discrete complex systems, 2014. https://doi.org/10.1007/978-3-319-18812-6_6) to develop computational methods that greatly reduce this computational cost. We give a detailed algorithm and apply it to (i) the lac operon model for Escherichia coli proposed by Veliz-Cuba and Stigler (2011), and (ii) the regulatory network involved in the control of the cell cycle and cell differentiation in the Caenorhabditis elegans vulva precursor cells proposed by Weinstein et al. (BMC Bioinform 16(1):1, 2015). In both cases, we uncover all possible limit cycle structures for these networks under sequential updates. Specifically, for the lac operon model, rather than examining all \(10! > 3.6 \cdot 10^6\) sequential update orders, we demonstrate that it is sufficient to consider 344 representative update orders, and, more notably, that these 344 representatives give rise to 4 distinct attractor structures. A similar analysis performed for the C. elegans model demonstrates that...
it has precisely 125 distinct attractor structures. We conclude with observations on the variety and distribution of the models’ attractor structures and use the results to discuss their robustness.

Keywords Discrete dynamical systems · Boolean networks · Update schedules · Sequential dynamical systems · Attractor structures · Long-term behavior · Enumeration · Classification

1 Introduction

Here we will assume that biological models are represented as maps of the form

\[ F = (f_1, \ldots, f_n) : K^n \rightarrow K^n, \]

where \( K \) is some suitable finite set like \( \{0, 1\} \), see, for example (Goles and Martínez 2013). We refer to maps of the form (1) as discrete graph dynamical systems (GDSs). Associated to \( F \), we have its dependency graph \( G \) with vertex set \( V(G) = \{1, 2, \ldots, n\} \) and edges all \( \{i, j\} \) for which the function \( f_j \) depends non-trivially on the vertex state \( x_i \), see (Adiga et al. 2016). The map (1) may be assembled through parallel update of states (the synchronous case), or through asynchronous update methods such as sequential or block sequential (Goles et al. 2013). With this system representation, questions about an underlying biological system can be turned into precise questions about the dynamical system defined by the map \( F \) in (1) using the structure of \( G \) and the properties of the maps \( f_i \).

In this paper, we focus on the attractors of the map in Eq. (1); they capture the possible long-term behaviors of a system model. In the context of a biological system, the attractors will often represent some physical observable such as the stages of a cell cycle. Naturally, it is desirable that the long-term dynamics of a model reflects the associated experimental observations.

In this paper, we demonstrate our mathematical techniques for exploring the possible attractor structures using the following biological network models: the regulatory network involved in the control of the cell cycle and cell differentiation in the Caenorhabditis elegans vulva precursor cells proposed by Weinstein et al. (2015), and the lac operon in Escherichia coli proposed by Veliz-Cuba and Stigler (2011). These models are examples of GDSs; more details surrounding these networks are presented in Sect. 4. The authors in both cases evaluated their models using a synchronous (or parallel) update method and found their models to exhibit dynamical behavior that is consistent with published experimental results.

For a synchronous model, the assumption is that all genes make a transition at the same time. While there is seldom enough kinetic information to discern a precise order of state transitions (Garg et al. 2008), the assumption of synchronized transitions may not be biologically realistic. By relying solely on analysis derived using a synchronous update scheme, one runs the risk of missing important insight into the structural stability of the model, see, for example (Demongeot et al. 2010). To assess how one’s model behaves under perturbations can offer valuable insight into its structural stabil-
ity which in turn can help justify the extent to which one’s conclusions are generic. For this reason, it is advantageous to examine how a biological model behaves under asynchronous update schemes such as sequential update orders. In general, obtaining this information across all sequential update orders is computationally intractable as here the number of updates grows as $n!$, where $n$ is the number of vertices in the model. Although existing programs, such as those in Müßel et al. (2010) and Řehůřek and Sojka (2010), can be used to compute the attractors from a sequential update, it would be extremely costly to perform a brute force analysis over all $n!$ sequential updates.

In this paper, we present an algorithm for analyzing the possible dynamics resulting from all the possible sequential update orders. Other update schemes are discussed in Sect. 7. Our analysis provides biological modelers immediate feedback on how their model behaves under different asynchronous updating orders and the diversity in attractor structures. Clearly, a large degree of attractor structure sensitivity with respect to the update scheme raises a cautionary flag regarding the validity and soundness of the modeling effort.

The concept of network robustness is presented in Kitano (2004) as a uniquitously observed and necessary property of biological systems. Naturally, one may also consider systems capable of evolution, in which robustness is present up to some point. The purpose of this paper is not to consider robustness versus evolution of the system being modeled, but instead to offer the modeler a systematic way to examine model robustness with respect to the update scheme. For this, we say that a dynamical property is robust when it is not affected by small perturbations. In a real system, these perturbations can model the inherent noise present in nature, whereas in a model, perturbations refer to changes in the state variables or to changes in the specification of the model itself (Aracena et al. 2009). In this paper, we will consider perturbations of the update order. Unlike perturbations to the system state, these modifications represent changes to the model itself and therefore may cause variations in the dynamics of the system. Here we only consider variations across different sequential (permutation) update orders. For more information on robustness of update orders, see (Macauley and Mortveit 2011b, 2013, 2014; Aracena et al. 2009).

Several biological network models have previously been studied in the context of update order robustness such as the lac operon model (Montalva et al. 2014), mammalian cell cycle network model (Ruz et al. 2014), and yeast cell cycle network (Goles et al. 2013). In these papers, the authors study update schedule robustness by analyzing dynamics properties such as long-term behavior (fixed points and limit cycles) and attractor basin sizes across all fixed deterministic updating schemes. We extend and complement this work by presenting computational algorithms for identifying all possible attractor structures that can be obtained for sequential update orders. At a high level, this algorithm is based on the body of work in Macauley and Mortveit (2009, 2014, 2013, 2011b) with key facts outlined below. It assumes that we have a fixed list of vertex functions $f = \{f_i\}_{i=1}^n$ capturing the local evolution with associated dependency graph $G$. All definitions are carefully presented in Sect. 2.
• For an update sequence \( \pi = (\pi_1, \ldots, \pi_n) \) of the vertices of \( G \), we obtain a map \( F_\pi \) of the form (1) by applying the maps \( f_i \) in the sequence specified by \( \pi \), see Eq. (4).

• From Reidys (1998) we have the equality \( F_\pi = F_\pi' \) whenever the acyclic orientations \( O_\pi \) and \( O_\pi' \) of \( G \) are equal. All possible maps of the form \( F_\pi \) can therefore be obtained from \( \text{Acyc}(G) \), the set of all acyclic orientations of \( G \): for each acyclic orientation \( O \), pick a sequence \( \pi \) compatible with \( O \) (a linear extension of \( O \)).

• From the main result in Macauley and Mortveit (2009) we know that the maps \( F_\pi \) and \( F_\pi' \) have the same attractor structure whenever the corresponding acyclic orientations \( O_\pi \) and \( O_\pi' \) are \( \kappa \)-equivalent, that is, are related by a sequence of source-to-sink operations. We remark that this is an equivalence relation.

• Using results from Macauley and Mortveit (2008), we show how to construct a complete set of update sequence representatives for \( \kappa \)-equivalence. From this set of representatives, we can generate all possible attractor structures of maps of the form \( F_\pi \).

• We also show how to compute \( \kappa(G) \), the number of \( \kappa \)-classes of \( G \). The quantity \( \kappa(G) \) is an upper bound for the number of possible attractor structures that can be generated, and it is typically orders of magnitude smaller than \( n! \), the total number of update orders \( \pi \). For example, in the lac operon network we have \( \kappa(G) = 344 \) which is four orders of magnitude less than \( 10! \).

The algorithm that we present is a concise, computational recipe for constructing a complete set of update sequence representatives for attractor equivalence. This is a prime example of a structure-to-function result: The theory and algorithm are based entirely on the combinatorial structures of the dependency graph \( G \).

**Significance** In our approach, we have coarsened the analysis to consider the variety and distribution of cycle structures across different sequential updates, rather than supplying full details about the complete dynamics of the model. This analysis may point to clear inconsistencies and instabilities resulting from the model choice. For example, a biological system may be known to only have steady states (fixed points). If a proposed model of a biological system exhibits limit cycles as attractors, it may be an indication that the model is not valid or has shortcomings. Our method helps reduce the computational burden related to conducting this and other analyses for asynchronous models.

**Paper outline** We present basic terminology and theory in Sect. 2 followed by a short overview of regulatory network models for *C. elegans* and lac operon in Sect. 4. Methods and algorithms are presented in Sect. 3 with computational illustrations presented in Sect. 5. Here we demonstrate our method by identifying all limit cycle structures in the *C. elegans* and lac operon network models in the case of sequential update orders. To the best of our knowledge, results pertaining to the former model have not been published, whereas partial results for the latter model have appeared in Montalva et al. (2014). Concluding remarks appear in Sect. 7.
2 Background, Terminology, and Notation

The theory covered in this section is a careful assembly of existing results that are needed for deriving our algorithms (Reidys 1998; Macauley and Mortveit 2014, 2011a, 2009). A sound understanding of these results is essential to using the algorithms and techniques effectively. We consider discrete dynamical systems as in Eq. (1) where each map $f_i$ is of the form $f_i: K^n \rightarrow K$. In general, each $f_i$ will only depend non-trivially on some subset of the vertex states $x_1$ through $x_n$, a fact captured by the dependency graph $G$ defined earlier. Each vertex $i$ of $G$ has assigned a vertex state $x_i \in K$ and a vertex function of the form $f_i: K^{d(i)+1} \rightarrow K$ taking as arguments the states of vertices in the closed 1-neighborhood of $i$ in $G$. Here $d(i)$ is the degree of vertex $i$. We write $n[i]$ for the ordered sequence of vertices from the 1-neighborhood of $i$ (note that $i$ is included), and $x[i]$ for the corresponding sequence of vertex states, see (Adiga et al. 2016). The system state is the $n$-tuple consisting of all the vertex states and is denoted by $x = (x_1, \ldots, x_n) \in K^n$.

If we apply the functions $(f_i)_i$ synchronously (in parallel), we obtain the graph dynamical system map $F$ as in Eq. (1) given by

$$F(x_1, \ldots, x_n) = (f_1(x[1]), \ldots, f_n(x[n])).$$

(2)

However, in this paper we will consider permutation update sequences, that is, we will apply the functions $f_i$ in the order given by a permutation $\pi = (\pi_1, \ldots, \pi_n)$. For this, we first introduce the notion of $G$-local functions (Kuhlman et al. 2011). Here the $G$-local function $F_i: K^n \rightarrow K^n$ is given by

$$F_i(x_1, \ldots, x_n) = (x_1, x_2, \ldots, f_i(x[i]), \ldots, x_n).$$

(3)

Using the permutation $\pi \in S_G$ (the set of all permutations of $V(G)$) as an update sequence, the corresponding sequential dynamical system (SDS) map $F_\pi: K^n \rightarrow K^n$ as in Eq. (1) is given by

$$F_\pi = F_{\pi_n} \circ F_{\pi_{n-1}} \circ \cdots \circ F_{\pi_1}.$$ 

(4)

The phase space of the dynamical system map $F$ in (1) is the directed graph $\Gamma(F)$ with vertex set $K^n$ and directed edges all pairs $(x, F(x))$. A state on a cycle in $\Gamma(F)$ is called a periodic point, and a state on a cycle of length one is called a fixed point (Kuhlman et al. 2011). The sets of all such points are denoted by Per($F$) and Fix($F$), respectively. The collection of all the cycles in the phase space represents all possible long-term dynamics of the system model.

**Definition 1** Let $F$ be as in Eq. (1). The cycle structure of $F$ is the unlabeled subgraph of $\Gamma(F)$ induced by Per($F$).

We represent the cycle structure of a phase space using multiset notation so that, e.g., $\{1(2), 2(3)\}$ denotes a phase space with two fixed points and three 2-cycles as in Example 1 and shown on the left side in Fig. 1.
Fig. 1 The graph $G$ of Example 1 is shown in the middle. The phase space of the map $F$ for the parallel update is shown on the left, while the phase space of the map $F_\pi$ obtained under a sequential update order $\pi = (1, 2, 3, 4)$ is shown on the right. Clearly, the update scheme matters. This figure is used with permission from Kuhlman and Mortveit (2014).

For illustrations, we will use Boolean bi-threshold vertex functions (Kuhlman and Mortveit 2014) $t_{i,k^\uparrow,k^\downarrow,m} : \{0,1\}^m \rightarrow \{0,1\}$ defined by

$$\begin{align*}
t_{i,k^\uparrow,k^\downarrow,m}(x_1, \ldots, x_m) =
\begin{cases}
1, & \text{if } x_i = 0 \text{ and } \sum_{j=1}^{n} x_j \geq k^\uparrow, \\
0, & \text{if } x_i = 1 \text{ and } \sum_{j=1}^{n} x_j < k^\downarrow, \\
x_i, & \text{otherwise},
\end{cases}
\end{align*}$$

(5)

where the integer parameters $k^\uparrow$ and $k^\downarrow$ are the up-threshold and down-threshold, respectively.

**Example 1** To illustrate the above concepts, let $G$ be the simple graph shown in the middle of Fig. 1 and let each vertex function be given by a bi-threshold function with thresholds $(k^\uparrow, k^\downarrow) = (1, 3)$. The (synchronous) map $F$ has the phase space shown on the left in Fig. 1. Using $\pi = (1, 2, 3, 4)$ as the update sequence, we obtain the map $F_\pi$ with phase space as shown on the right in Fig. 1. This example is taken from Kuhlman and Mortveit (2014).

The dependency graph $G$ is generally directed and will contain loops (edges of the form $(v, v)$.) Let $G_c$ be the graph obtained from $G$ by (i) removing all self-loops, (ii) replacing each directed edge by an undirected edge, and (iii) collapsing all parallel edges to a single edge. For all the theory in this paper, we obtain exactly the same results and conclusions whether we use $G$ or $G_c$. To see this, assume that $G$ is directed and possibly also contains loops. The questions of determining whether $F_\pi$ and $F_\pi'$ over $G$ are functionally- or cycle-equivalent are precisely captured by the graph $G_c$ as before: consider the two compositions $F_a \circ F_b$ and $F_b \circ F_a$ in the case of functional equivalence. The conditions for when these compositions are equal are independent of edges in $G$ being directed or not, or if there are loops. We simply state that this argument transfers to and holds for $\kappa$-equivalence, but refer the reader to the proof for cycle equivalence of $F_\pi$ under cyclic shifts of the update sequence given in Macauley and Mortveit (2009). The upshot of all this is that for a given biological model, we may simply consider the $G_c$ versions of their respective networks. When referring to these graphs later in this paper, we will not distinguish between $G$ and $G_c$; it will be clear from the context. We call $G_c$ the combinatorial graph induced by $G$. 
2.1 Functional Equivalence

The first step to address is: for update sequences $\pi$ and $\pi'$, when is $F_\pi = F_{\pi'}$? To answer this, Reidys (1998) introduced the equivalence relation $\sim_\alpha$ on $S_G$ by $\pi \sim_\alpha \pi'$ if $\pi$ can be transformed into $\pi'$ by a sequence of adjacent transpositions of vertices $v$ and $v'$ where $\{v, v'\}$ is not an edge in $G$. It is clear that $\pi \sim_\alpha \pi'$ implies the functional equality $F_\pi = F_{\pi'}$, see (Macauley and Mortveit 2011a). The equivalence classes are called $\alpha$-classes, the $\alpha$-class containing $\pi$ is denoted by $[\pi]_G$, and the collection of all $\alpha$-classes is written $S_G / \sim_\alpha$.

Although it may be natural to address functional equivalence through $\sim_\alpha$, it is computationally more efficient to use the acyclic orientations of $G$. An orientation of $G$ is an assignment of direction to each edge $e \in E(G)$; this can be represented by a map $O_G : E(G) \longrightarrow V(G) \times V(G)$; the orientation $O_G$ is acyclic if this assignment contains no cycles. The set of acyclic orientations of $G$ is denoted $\text{Acyc}(G)$, and we set $\alpha(G) = |\text{Acyc}(G)|$. Here is the connection: A permutation $\pi \in S_G$ defines an acyclic orientation $O(\pi)$ by $O(\{i, j\}) = (i, j)$ if $i$ precedes $j$ in $\pi$ and $O(\{i, j\}) = (j, i)$ otherwise, see (Macauley and Mortveit 2011a) and Fig. 2, left. Moreover, from Reidys (1998) we have a bijection between the $\alpha$-classes of $G$ and the acyclic orientations of $G$ given by

$$
\phi_G : S_G / \sim_\alpha \longrightarrow \text{Acyc}(G), \quad \phi_G([\pi]_G) = O(\pi).
$$

(6)

The number of equivalences classes under $\sim_\alpha$ therefore equals $\alpha(G)$, and it follows that the measure $\alpha(G)$ is an upper bound for the number of functionally distinct maps $F_\pi$ that can be generated by varying the update order.

**Fact 1. Testing for equality of sequential GDS maps** We can test if $\pi$ and $\pi'$ are in the same $\alpha$-class, which would imply $F_\pi = F_{\pi'}$, by computing and comparing $O(\pi)$ and $O(\pi')$. The complexity of this test is $O(m)$ where $m = |E(G)|$.

**Fact 2. Bounding the number of distinct sequential GDS maps** We can compute this bound $\alpha(G)$ through the recursion relation

$$
\alpha(G) = \alpha(G/e) + \alpha(G \setminus e), \quad e \in E(G).
$$

(7)

Here $G/e$ is the graph constructed from $G$ by contracting the edge $e$, and $G \setminus e$ is the graph obtained from $G$ by deleting the edge $e$.

**Example 2** For the graph $G$ in Example 1, we have $\alpha(G) = 18$. To see this, take $e = \{1, 3\}$ so that $\alpha(G/e) = 4$ and $\alpha(G \setminus e) = 14$. If we take $\pi = (1, 2, 3, 4)$, we get the acyclic orientation shown on the left in Fig. 2.

2.2 Cycle Equivalence

Two dynamical systems of the form (1) are cycle equivalent if they have the same cycle structure. Here we specialize this to the case where we have update sequences $\pi$...
Fig. 2 For the graph $G$ in Example 1 with $\pi = (1, 2, 3, 4)$, the orientation $O(\pi)$ is shown on the left (vertex 1 is the unique source). On the right, the acyclic orientation of the shifted order $\sigma(\pi) = (2, 3, 4, 1)$ is shown. Notice these two orientations are click-related as we have converted vertex 1 from a source to a sink and $\pi'$ and corresponding maps $F_\pi$ and $F_{\pi'}$. First, define the cyclic shift $\sigma(\pi)$ by $\sigma(\pi) = (\pi_2, \pi_3, \ldots, \pi_n, \pi_1)$. At the level of acyclic orientations, we see that mapping $\pi$ to $\sigma(\pi)$ corresponds precisely to converting $\pi_1$ from a source in $O(\pi)$ to a sink in $O(\sigma(\pi))$, see Fig. 2. Following (Macauley and Mortveit 2011a), we call the conversion of a source vertex to a sink vertex in $O \in \text{Acyc}(G)$ a source-to-sink operation, or a click. Two acyclic orientations $O, O' \in \text{Acyc}(G)$ where $O$ can be transformed into $O'$ by a sequence of clicks are said to be click-related. The transitive and reflexive closure of this relation is denoted by $\sim_\kappa$; it is an equivalence relation, see (Macauley and Mortveit 2011a). Its equivalence classes are called $\kappa$-classes, are denoted by $[O]_G$, and the set of all $\kappa$-classes is $\text{Acyc}(G)/\sim_\kappa$.

We introduced $\kappa$-equivalence in Macauley and Mortveit (2009) because $O(\pi) \sim_\kappa O(\pi')$ implies that the maps $F_\pi$ and $F_{\pi'}$ are cycle-equivalent. How do we efficiently determine if $O(\pi) \sim_\kappa O(\pi')$? For this, let $C$ be a directed simple cycle in $G$ of length $k$, that is, $C = (v_0, v_1, \ldots, v_{k-1}, v_0)$ where $\{v_{i-1}, v_i\} \in E(G)$ for $i = 1, \ldots, k$. Define the (scalar) Coleman $\nu$-function (Coleman 1989)

$$\nu_C : \text{Acyc}(G) \rightarrow \mathbb{Z},$$

(8)

by $\nu_C(O) = n^+ - n^-$ where $n^+$ is the number of edges in $C$ oriented along $O$ and $n^-$ is the number of edges in $C$ oriented opposite of $O$, see (Macauley and Mortveit 2011a). We need to extend $\nu_C$ to cover all the cycles of $G$, and for this we pick a cycle basis $\mathcal{C} = (C_1, \ldots, C_m)$ and define $\nu_C : \text{Acyc}(G) \rightarrow \mathbb{Z}^m$ by

$$\nu_C = (\nu_{C_1}, \ldots, \nu_{C_m}),$$

(9)

see (Macauley and Mortveit 2011a), where it is also shown that $\nu_C$ is a complete invariant for $\kappa$-equivalence. In other words, $\nu_C(O) = \nu_C(O')$ if and only if $O \sim_\kappa O'$. We summarize this and more in the following facts that form the basis for the algorithm presented in Sect. 3.

Fact 3. Testing for cycle equivalence of sequential GDS maps To determine if $\pi$ and $\pi'$ are $\kappa$-equivalent, and therefore if $F_\pi$ and $F_{\pi'}$ are cycle-equivalent, (i) pick
a cycle basis $C$ of $G$; (ii) form $O(\pi)$ and $O(\pi')$, and (iii) compare $\nu_C(O(\pi))$ and $\nu_C(O(\pi'))$.

**Fact 4. Exhaustive exploration of all possible attractor structures for sequential GDS maps** Let $\text{Acyc}_v(G)$ denote the set of all acyclic orientations of $G$ where $v$ is the unique source, and let $f = (f_i)_i$ be fixed vertex functions. To exhaustively explore all possible cycle structures for maps of the form $F_\pi$, we use the fact that for any $v \in V(G)$ the set $\text{Acyc}_v(G)$ is a complete set of $\kappa$-equivalence class representatives, see (Macauley and Mortveit 2009). The corresponding set of update sequence representatives is obtained by selecting precisely one linear extension for each element of $\text{Acyc}_v(G)$.

**Fact 5. Bounding the number of possible attractor structures for sequential GDS maps** The measure $\kappa(G) = |\text{Acyc}(G)/\sim_\kappa|$ is an upper bound for the number of distinct cycle structures that can be obtained by varying the update sequence $\pi$. We can evaluate $\kappa(G)$ in a manner similar to $\alpha(G)$, that is,

$$\kappa(G) = \kappa(G/e) + \kappa(G \setminus e), \quad e \in E(G) \text{ a cycle edge.} \quad (10)$$

If $G$ is cycle-free, then $\alpha(G) = 1$. In particular, if $G$ is a tree, there is only one cycle structure.

**Example 3** Continuing the previous example sequence, we see that $\kappa(G) = 4$. As before, take $e = \{1, 3\}$. Then $\kappa(G/e) = 1$ and $\kappa(G \setminus e) = 3$. One cycle basis for $G$ is $C = \{C_1 = (1, 3, 4, 1), C_2 = (1, 2, 3, 1)\}$. For example, $\nu_C(O(\pi = (1, 2, 3, 4))) = (1, 1)$, while $\nu_C(O(\pi' = (4, 3, 2, 1))) = (-1, -1)$. We conclude that $\pi$ and $\pi'$ are not $\kappa$-equivalent. However, we note that $F_\pi$ and $F_{\pi'}$ may still be cycle-equivalent: That will depend on the particular choice of functions $(f_i)_i$, a fact that motivates the next definition.

**Definition 2** Let $(f_i)^n_{i=1}$ be a sequence of vertex functions as above with dependency graph $G$. Two acyclic orientations $O(\pi), O(\pi') \in \text{Acyc}(G)$ are cycle-equivalent if $F_\pi$ and $F_{\pi'}$ are cycle-equivalent. Cycle equivalence classes are denoted $[O(\pi)]_F$, and we let $\kappa_F(G)$ denote the total number of cycle equivalence classes.

Clearly, cycle equivalence is a coarsening of $\kappa$-equivalence. By construction it follows that $\kappa_F(G) \leq \kappa(G)$. The following is an open question: for a given (but arbitrary) graph $G$, are there vertex functions $(f_i)_i$ with dependency graph $G$ such that $\kappa_F(G) = \kappa(G)$?

Again $\alpha$- and $\kappa$-equivalence are strictly combinatorial constructions based on the structure of the dependency graph $G$. However, they govern and restrict dynamics of corresponding GDS maps and significantly reduce the computational burden of many analysis problems. For example, to evaluate $\kappa_F(G)$ we only need to determine the cycle structure across a set of $\kappa$-class representatives as in Fact 4. To get a sense of impact, consider the following example.

**Example 4** Let $G$ be the binary hypercube of dimension 3 and consider some fixed, but arbitrary sequence of vertex functions. In this case, there are $8!$ possible update
sequences, we have \( \alpha(G) = 1862 \) and \( \kappa(G) = 133 \). To study cycle equivalence over \( G \), the above theory allows us to simplify the analysis by a factor of \( 8!/133 > 300 \). If the complete analysis over all \( 8! \) update sequences would take 1 day, our approach would complete in less than 5 minutes.

3 Overview of Method

Algorithm 1 generates a complete set of \( \kappa \)-class representatives for a graph \( G \). Note that a linear extension of an acyclic orientation \( O \) is a permutation \( \pi \) for which \( O_G(\pi) = O \). Using this set of representatives, the exploration of the possible cycle structures that can arise through varying the permutation update sequence now becomes more compelling. The algorithm uses Fact 4. Note that we choose a vertex \( v \) of maximal degree as that reduces the remaining computations. Again, we assume that \( G \) has been converted into the combinatorial graph \( G_c \) if \( G \) is not simple. Please note that the Python code and associated library that were used for the subsequent computations are available for download, see Sect. 6 for details.

### Algorithm 1

Complete set of \( \kappa \)-class representatives.

1: Select \( v \in V(G) \) of maximal degree
2: \( G' := G \setminus v \)
3: \( \text{kappaClassPermutationReps} := \emptyset \)
4: for \( O \in \text{AcyclicOrientations}(G') \) do
5: flag = False
6: for \( v' \) in \( V(G') \) do
7: if \( \{ v', v \} \notin E(G) \) & \( v' \in \text{SourceVertices}(O) \) then
8: flag = True
9: break
10: if not flag then
11: \( \text{kappaClassPermutationReps.Insert( Concatenate( v, \text{LinearExtension}(O) ) )} \)
12: return \( \text{kappaClassPermutationReps} \)

We illustrate the algorithm using Example 1 from Sect. 2 for reference. Here the dependency graph \( G \) shown in Fig. 1 is simple. To construct all the acyclic orientations of \( G \) where \( v \) is a unique source, that is, \( \text{Acyc}_v(G) \), form the graph \( G' \) (delete \( v \) from \( G \)) and start by constructing all its possible orientations (there will be \( 2^N \) of these, where \( N \) is the number of edges of \( G' \)); discard the ones that are cyclic. For this, a graph library that can perform a topological sorting of directed graphs, e.g., NetworkX (Hagberg et al. 2008), can be very useful. If an orientation is cyclic, the topological sort will fail and the orientation can be discarded. However, if the orientation \( O' \) is acyclic and contains no source vertex \( v' \) such that \( \{ v, v' \} \notin E(G) \), we form \( O \in \text{Acyc}_v(G) \) by adding \( v \) to \( O' \) and reintroducing all deleted edges incident to \( v \) orienting them so that \( v \) is a source. From a corresponding collection of update sequence representatives, we can construct the cycle structures of the phase spaces \( \Gamma(F_\pi) \).

The algorithm returns a complete set of \( \kappa \)-class representatives represented by update orders. If we compute \( F_\pi \) for all representative update orders \( \pi \) in this col-
Fig. 3 The vertex functions for the network model in the lac operon

\[
f_1 = x_4 \land \neg x_5 \land \neg x_6 \\
f_2 = x_1 \\
f_3 = x_1 \\
f_4, \mu_0 = \neg \mu_0 \\
f_5 = \neg x_7 \land \neg x_8 \\
f_6 = (\neg x_7 \land \neg x_8) \lor x_5 \\
f_7 = x_9 \land x_3 \\
f_8 = x_9 \lor x_{10} \\
f_{9, \mu_0, \mu_1} = x_2 \land \mu_1 \land \neg \mu_0 \\
f_{10, \mu_0, \mu_1, \mu_2} = ((\mu_2 \land x_2) \lor \mu_1) \land \neg \mu_0
\]

Fig. 4 The list of vertex functions for the network model of C. elegans

In this section, we present the biological network models that will be used in the computational analysis. In this biological context, the vertex functions, which are listed in Figs. 3 and 4, represent regulatory interactions between molecules. The dependency graphs for these networks seen in Figs. 5 and 6 are directed and may contain loops. For simplicity, we have relabeled the vertices.

4 Biological System Models

In this section, we present the biological network models that will be used in the computational analysis. In this biological context, the vertex functions, which are listed in Figs. 3 and 4, represent regulatory interactions between molecules. The dependency graphs for these networks seen in Figs. 5 and 6 are directed and may contain loops. For simplicity, we have relabeled the vertices.

4.1 Network Model For Lac Operon

The lac operon in Escherichia coli is the system responsible for the metabolism of lactose in the absence of glucose and is known to exhibit bistability, in the sense
that the operon is either induced (ON) or uninduced (OFF) (Montalva et al. 2014). In Veliz-Cuba and Stigler (2011) the authors proposed a Boolean model for it, see Fig. 5.

Here \( \mu_0 \in \{0, 1\} \) and the pair \((\mu_1, \mu_2)\) can be in three states: low, medium, or high which is represented by \((\mu_1, \mu_2) = (0, 0), (0, 1), \) and \((1, 1)\), respectively.

The authors in Montalva et al. (2014) consider dependency graph \( G \) under a parallel update and only find fixed points which represent the steady ON or OFF states of the operon. For more information on the biological interpretations of this network model, see (Veliz-Cuba and Stigler 2011). We note that comprehensive analyses of this model under all sequential updates have been studied previously by Goles et al. in (Montalva et al., 2014).

### 4.2 Network Model in Caenorhabditis elegans

Here we introduce the regulatory network model involved in the control of the cell cycle and cell differentiation in the \( C.\) elegans vulva precursor cells proposed in Weinstein et al. (2015). This model is, to our knowledge, the first model to include the molecular mechanism involved in the control of the postembryonic cell cycle of \( C.\) elegans.

In Fig. 6, we show the dependency graph \( G \). We denote the parameters by \( \mu_0, \mu_1 \) where \( \mu_0 \in \{0, 1, 2, 3\} \) and \( \mu_1 \in \{0, 1\} \).

The larger graph \( G' \) is the result of extending \( G \) to include the parameters/vertices \( \mu_0, \mu_1 \) and all their incident dashed edges. For the system over \( G' \), we write \( f_{3, \mu_0, \mu_1}, f_{1, \mu_1} \) to \( f_3, f_1 \) and let the parameter vertex functions \( f_{12} \) and \( f_{13} \) simply be the identity functions returning their own state. As a consequence of this, the phase space of \( F \) can then be obtained as the disjoint union of the eight phase spaces.
of \((F_{\mu_0,\mu_1})_{\mu_0,\mu_1}\) over \(G\). To assess dynamical diversity and equivalences, we therefore have two graphs to consider, \(G\) and \(G'\). The advantage of using parameters is to cut down on memory requirements and allow for a compact discussion of results in Sect. 5.

The authors in Weinstein et al. (2015) consider the dependency graph \(G'\) under a parallel update and interpret periodic \(n\)-cycles found as the patterns of molecular activation of the three vulval fates that cycle through the cell cycle. For more information on vulval fates or the cell differentiation process, see (Weinstein et al. 2015).

## 5 Results

The dependency graphs for the biological network models seen in Figs. 5 and 6 are directed and contain loops. However, as described in Sect. 2, we consider the combinatorial graph obtained by removing all loops and replacing each edge \((a, b)\) and pairs of edges \((a, b), (b, a)\) by an undirected edge \(\{a, b\}\).

### 5.1 Network Model for lac Operon

The possible dynamics of this network model have already been studied comprehensively in Montalva et al. (2014). In the following, we will only consider the case where \(\mu_0 = \mu_1 = 0\) and \(\mu_2 = 1\), as it is shown in Montalva et al. (2014) that all other parameter and update schedule configurations reach a single fixed point.

We extend and complement this work by utilizing the \(\kappa\)-class representative method to explicitly list all possible long-term dynamical structures of \(F_{(0,1)}\) under sequential updates using the dependency graph \(G\), see Table 1. The two steady states (fixed points) found are \((0, 0, 0, 1, 1, 1, 0, 0, 0, 0)\) and \((1, 1, 1, 1, 0, 0, 0, 1, 0, 1)\), which correspond to the operon being OFF and ON, respectively.

Using Algorithm 1, we find \(\omega(G) = 14112\) and \(\kappa(G) = 344\). We compute the phase space for each of these 344 representative updates and find that \(\kappa_{F_{(0,1)}}(G) = 4\). The efficiency of pre-computing \(\kappa\)-classes is clear here, as we can extract cycle structure results using only 344 update order representatives instead of \(10!\) or even 14112. The low ratio \(\kappa_{F_{(0,1)}}(G)/\kappa(G)\) provides an initial measurement on the update order robustness of the network. A low ratio would suggest a high degree of cycle structure preservation across different update permutations, a general characteristic of network robustness.

| Cycle structure multiset | frequency of \(\kappa\)-classes in cycle equivalence class |
|--------------------------|----------------------------------------------------------|
| \{1(2)\} : 263          | \{1(2), 2(1)\} : 31                                      |
| \{1(2), 3(2)\} : 31     | \{1(2), 2(1), 4(3)\} : 19                               |

Table 1 The results displayed represent the cycle structure multisets for all \(\kappa\)-classes using the fixed parameters \(\mu_0 = \mu_1 = 0\) and \(\mu_2 = 1\).
5.2 Network Model in Caenorhabditis elegans

In the following, we consider the two dependency graphs of this network model denoted by $G$ and $G'$, see Sect. 4. Here we show how to relate $\alpha$- and $\kappa$-classes over these two graphs.

For $\alpha$-equivalence, we note that each acyclic orientation $O(\pi) \in \text{Acyc}(G)$ extends to precisely six acyclic orientations $O_1(\pi_1), \ldots, O_6(\pi_6) \in \text{Acyc}(G')$ where all maps $O_i(\pi_i)$ are equivalent when restricted to $E(G)$. Consequently, $\alpha(G') = 6\alpha(G)$. However, all linear extensions $\pi_i$ from these six orientations result in equivalent dynamics under $F_{\pi_i}$.

For $\kappa$-equivalence, we use Coleman’s $v$-function (see Sect. 2). Let $C = (C_1, \ldots, C_m)$ be a cycle basis for $G$. We extend $C$ to a cycle basis for $G'$ by adding $C'$ to get $C' = (C_1, \ldots, C_m, C')$, see Fig. 6. Since $v_C(O') \in \{-1, 1\}$ for all $O' \in \text{Acyc}(G')$, it follows that each $\kappa$-class over $G$ extends to precisely two $\kappa$-classes over $G'$. Consequently, $\kappa(G') = 2\kappa(G)$. The six orientations of $G$ seen in the paragraph above are distributed across two $\kappa$-classes over $G'$ according to their value under $v_C$. However, due to the functional equivalence of the six orientations, these two distinct $\kappa$-classes must be contained in the same cycle equivalence class over $G'$.

We conclude that each $\kappa$-class over $G$ gives rise to two distinct and cycle-equivalent $\kappa$-classes over $G'$ and each $\alpha$-class over $G$ gives rise to six distinct and functionally equivalent $\alpha$-classes over $G'$. To find all possible cycle structures using $G'$, it is sufficient to evaluate each $\kappa$-class representative using $G$ under all eight parameter configurations, see Sect. 4.

We first consider dynamics over the dependency graph $G$ and find $\alpha(G) = 158208$ and $\kappa(G) = 5312$ using Algorithm 1. These are graph combinatorial measures which are independent of the parameter choice and functions; however, altering the parameters $\mu_0$ and $\mu_1$ may change the dynamics of the system and therefore change the measure $\kappa_{F_{\mu_0,\mu_1}}(G)$, see Table 2.

**Observation 1** Let $\pi \in S_G$ and parameters $(\mu_0, \mu_1) \in \{0, 1, 2, 3\} \times \{0, 1\}$ be given and fixed. The resulting phase space $\Gamma(F_{(\mu_0,\mu_1),\pi})$ is restricted to the following long-term dynamics:

1. A single n-cycle, or
2. Two distinct n-cycles with the same length n (bistability).

In both cases, we have $3 \leq n \leq 10$. If $(\mu_0, \mu_1) \in \{(0, 1), (1, 0), (1, 1), (3, 1)\}$ or a parallel update is used, we only obtain a single n-cycle (case 1).

In Table 2, we fix the parameters $(\mu_0, \mu_1)$ and for each combination, we display $\kappa_{F_{\mu_0,\mu_1}}(G)$ and the frequency of $\kappa$-classes that resulted in bistable cycle structures (two distinct n-cycles with the same length $n$). We note that with the use of a parallel update, we obtain a single n-cycle for any parameter combination. (Bistability is not present.)

For the extended dependency graph $G'$, we find $\alpha(G') = 949248$, $\kappa(G') = 10624$, and $\kappa_{F}(G') = 125$. In Table 3, we showcase some of the variety of cycle structures from $G$ by listing several cycle structure multisets found. The multisets found in Table 3 correspond to the cycle structures of some 4394
For each fixed parameter combination \((\mu_0, \mu_1)\), we display \(\kappa_{F_{\mu_0,\mu_1}}(G)\) and the frequency of \(\kappa\)-classes that result in bistable cycle structures.

| \((\mu_0, \mu_1)\) | \(\kappa_{F_{\mu_0,\mu_1}}(G)\) | Frequency of \(\kappa\)-classes that result in bistability |
|-------------------|-------------------------------|--------------------------------------------------|
| (0,0)             | 14                            | 1695                                             |
| (0,1)             | 8                             | 0                                                |
| (1,0)             | 8                             | 0                                                |
| (1,1)             | 8                             | 0                                                |
| (2,0)             | 12                            | 1664                                             |
| (2,1)             | 8                             | 84                                               |
| (3,0)             | 12                            | 1664                                             |
| (3,1)             | 7                             | 0                                                |

In each case there are 5312 total \(\kappa\)-classes.

The results displayed represent the cycle structure multisets for 4394 of the 10624 total \(\kappa\)-classes over \(G'\).

| Cycle structure multiset | Frequency of \(\kappa\)-classes in cycle equivalence class |
|-------------------------|----------------------------------------------------------|
| {3(11)}                | : 1094                                                   |
| {4(11)}                | : 524                                                    |
| {7(9)}                 | : 118                                                    |
| {6(9)}                 | : 132                                                    |
| {4(4), 5(4)}           | : 914                                                    |
| {5(4), 6(4)}           | : 662                                                    |
| {4(5), 5(4)}           | : 204                                                    |
| {3(6), 7(5)}           | : 196                                                    |

\(\kappa\)-classes. The frequency of a cycle structure multiset is the number of \(\kappa\)-classes that resulted in the same cycle structure displayed by the multiset. We can then group the \(\kappa\)-classes that resulted in the same cycle structure to form a cycle equivalence class. Statistics of multisets over all 10624 \(\kappa\)-classes are found in Table 4.

To gain further initial insight into the update robustness of the model, we may visualize how acyclic orientations are distributed across cycle equivalence classes. In Fig. 7, we enumerate all cycle equivalence classes and plot, in decreasing order, the percentage of total acyclic orientations contained in a corresponding class. In a robust network model, we expect this graph to be heavily front-loaded, as we expect a large percentage of possible dynamics to result in the same cycle structures.

We find that around 75% of all acyclic orientations are distributed across 23 cycle equivalence classes. To further discuss results in the context of system robustness, let us consider the cycle structure results found in Weinstein et al. (2015) when the update is a parallel one. The cycle structure multiset for the parallel update is \([9(4), 10(2), 11(2)]\). If we assume the model has robustness across update order variations, we expect cycle structure multiset data to remain similar in the context of the biological network. Finding and interpreting these similarities can require a vast amount of insight into the biological system.

In Table 4, we see that the total number of cycles within a cycle structure multiset is roughly preserved from the parallel update scheme, as 61.1% of all \(\kappa\)-classes result...
Fig. 7 Distribution of all acyclic orientations of $G'$ across all cycle equivalence classes. On the horizontal axis, we enumerate all 125 cycle equivalence classes and plot, in decreasing order, the percentage of total acyclic orientations contained in a corresponding class.

Table 4 The frequency of $\kappa$-classes that result in cycle structures with the same total number of $n$-cycles (same multiset size)

| Total number of cycles (size of multiset) | Frequency of $\kappa$-classes |
|-----------------------------------------|------------------------------|
| 8 : 6496                                 | 9 : 800                      |
| 10 : 570                                 | 11 : 2758                    |

in multisets of size eight. We also find that within a multiset, there are at most three distinct $n$-cycle lengths where the length $n$ is at least three and at most ten.

6 Software

The software used and referenced in this paper is available from GitHub (https://github.com/HenningMortveit/gds-framework-python) under the Artistic License 2.0 (https://opensource.org/licenses/Artistic-2.0). The source has been developed and tested under Python version 2.7. All computational examples in the text can be found in ./biographs-paper.py. By inspecting the system models in ./gds/biographs.py, one can adapt these to other systems and analyses.

7 Conclusions and Future Work

In this paper, we demonstrated a computationally efficient method for finding all possible cycle structures arising in asynchronous biological models as in Equation 4. In summary, we utilized $\kappa$-equivalences over the dependency graph $G$ to precompute classes of permutations whose corresponding SDS maps are cycle-equivalent. Therefore, computation over one representative update schedule per $\kappa$-class is sufficient for
comprehensive analysis of possible cycle structures. This analysis provides immediate feedback to biological modelers about the presence and diversity of possible long-term dynamical structures resulting from varying the update order and the robustness of their model. For example, in the network model of *C. elegans* we saw that the model is bistable under selected sequential updates and fixed parameters. Additionally, in the *lac* operon model we find that certain $n$-cycles are admissible. In the context of this biological model, these limit cycles have no clear interpretation. However, from the frequency distribution of $\kappa$-classes in Table 1, we see that these cycles are relatively uncommon.

Aside from efficiency, this method also provides a more refined approach to answering the general question of determining how the dynamics of a model change over different updating schemes. In this paper, we have simplified the question to determining the variety and distribution of cycle structures across different sequential updates. We are able to compare the total number of distinct cycle structures of a model, that is $\kappa F(G)$, based on a worse-case scenario arising from the bound $\kappa(G)$. In the case of overall attractor stability, we expect this ratio $\kappa F(G)/\kappa(G)$ to be small and expect a small variety of possible cycle structures. We see that this is clearly the case for the network model in the *lac* operon, as $10! = 3628800$ permutation update order dynamical systems (using $\mu_0 = \mu_1 = 0$ and $\mu_2 = 1$) will result in only four possible cycle structures. We see a similar case for the network model in the *C. elegans*, as $13! = 6227020800$ permutation update orders will result in 125 possible cycle structures.

One future direction of work would be to analyze the possible periodic states within the long-term dynamical structures. The results of such analysis may be better interpreted in the context of the biological system model and could provide additional feedback to modelers. For example, we may use this feedback to further investigate and interpret the cases where the network model in *C. elegans* is bistable. We claim that comprehensive information of this type is also efficiently achievable in an approach that utilizes $\kappa$-equivalence.

In this work, we have restricted ourselves to asynchronous updates using permutations sequences. In ongoing work, we are extending this theory to $\kappa$-equivalence for block-sequential update orders. While this adds complexity, exploring stability of biological models in this more general setting may also allow additional insight into sensitivity and robustness.

Finally, we mention that frequency of $\kappa$-classes, or more precisely, the frequency of cycle structures (as multisets) observed across $\kappa$-classes, gives initial insight into the respective likelihoods of attractors. A more elaborate and computationally demanding analysis may consider the following: Each $\kappa$-class is a set of acyclic orientations, and each acyclic orientation is associated with a set of linear extensions, that is, the update sequences that map to that particular orientation using Eq. (6). For a particular biological system, there may be a statistical distribution across the set of permutation update sequences, and one may choose to use this when inducing weights of the $\kappa$-classes. We have chosen to leave this analysis for future work.
Acknowledgements  We thank our external collaborators and members of the Network Systems Science & Advanced Computing (NSSAC) division for their suggestions and comments. This work has been partially supported by DTRA Grant HDTRA1-11-1-0016.

References

Adiga A, Galyean H, Kuhlman CJ, Levet M, Mortveit HS, Wu S (2016) Activity in Boolean networks. Nat Comput 16:1–13
Aracena J, Goles E, Moreira A, Salinas L (2009) On the robustness of update schedules in Boolean networks. Biosystems 97(1):1–8
Coleman A (1989) Killing and the Coxeter transformation of Kac–Moody algebras. Invent Math 95(3):447–477
Demongeot J, Goles E, Morvan M, Nouné M, Sené S (2010) Attraction basins as gauges of robustness against boundary conditions in biological complex systems. PLoS One 5(8):e11,793. https://doi.org/10.1371/journal.pone.0011793
Garg A, Di Cara A, Xenarios I, Mendoza L, De Micheli G (2008) Synchronous versus asynchronous modeling of gene regulatory networks. Bioinformatics 24(17):1917–1925
Goles E, Martínez S (2013) Neural and automata networks: dynamical behavior and applications, vol 58. Springer, Berlin
Goles E, Montalva M, Ruz GA (2013) Deconstruction and dynamical robustness of regulatory networks: application to the yeast cell cycle networks. Bull Math Biol 75(6):939–966
Hagberg A.A, Schult D.A, Swart P .J (2008) Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy2008). Pasadena, CA, USA, pp 11–15
Kitano H (2004) Biological robustness. Nat Rev Genet 5(11):826–837
Kuhlman CJ, Mortveit HS (2014) Attractor stability in nonuniform Boolean networks. Theor Comput Sci 559:20–33
Kuhlman CJ, Mortveit HS, Murrugarra D, Kumar V.A (2011) Bifurcations in Boolean networks. In: Discrete mathematics and theoretical computer science proceedings of the 17th international workshop on cellular automata and discrete complex systems (DMTCS Automata 2011), pp 29–46
Macauley M, Mortveit HS (2008) On enumeration of conjugacy classes of Coxeter elements. Proc Am Math Soc 136(12):4157–4165. https://doi.org/10.1090/S0002-9939-09-09884-0. arXiv:Math.CO/0711.1140
Macauley M, Mortveit HS (2009) Cycle equivalence of graph dynamical systems. Nonlinearity 22(2):421
Macauley M, Mortveit H.S (2011a) Posets from admissible Coxeter sequences. Electron J Comb 18(P197). Preprint: arXiv:math.DS/0910.4376
Macauley M, Mortveit HS (2011b) Update sequence stability in graph dynamical systems. Discret Contin Dyn Syst 4(6):1533–1541. https://doi.org/10.3934/dcdss.2011.4.1533 Preprint: arXiv:math.DS/0909.1723
Macauley M, Mortveit HS (2013) An atlas of limit set dynamics for asynchronous elementary cellular automata. Theor Comput Sci 504:26–37. Discrete Mathematical structures: from dynamics to complexity—DISCO 2011 24–26 November, 2011, Santiago, Chile. https://doi.org/10.1016/j.tcs.2012.09.015
Macauley M, Mortveit HS (2014) Cycle equivalence of finite dynamical systems containing symmetries. In: Isokawa T, Imai K, Matsui N, Peper F, Umeo H (eds) Cellular automata and discrete complex systems: 20th international workshop, AUTOMATA 2014, Himeji, Japan, July 7–9, 2014, Revised Selected Papers, Lecture Notes in Computer Science, vol 8996, pp 70–82. https://doi.org/10.1007/978-3-319-18812-6_6
Montalva M, Ruz GA, Goles E (2014) Attraction basins in a lac operon model under different update schedules. In: ALIFE 14: The 14th conference on the synthesis and simulation of living systems 14:689–690
Müssel C, Hopfensitz M, Kestler HA (2010) Boolnet: an R package for generation, reconstruction and analysis of boolean networks. Bioinformatics 26(10):1378–1380
Rehůřek R, Sojka P (2010) Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks. ELRA, Valletta, Malta, pp 45–50. http://is.muni.cz/publication/884893/en

Reidys C (1998) Acyclic orientations of random graphs. Adv Appl Math 21(2):181–192

Ruz GA, Goles E, Montalva M, Fogel GB (2014) Dynamical and topological robustness of the mammalian cell cycle network: a reverse engineering approach. Biosystems 115:23–32

Veliz-Cuba A, Stigler B (2011) Boolean models can explain bistability in the lac operon. J Comput Biol 18(6):783–794

Weinstein N, Ortiz-Gutiérrez E, Muñoz S, Rosenblueth DA, Álvarez-Buylla ER, Mendoza L (2015) A model of the regulatory network involved in the control of the cell cycle and cell differentiation in the caenorhabditis elegans vulva. BMC Bioinform 16(1):1

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Henning S. Mortveit1 · Ryan D. Pederson2

Henning S. Mortveit
Henning.Mortveit@virginia.edu

1 Engineering Systems and Environment and Network Systems Science & Advanced Computing, University of Virginia, Charlottesville, VA, USA

2 Department of Physics, University of California, Irvine, CA, USA