# DSGRN: Examining the Dynamics of Families of Logical Models

*Bree Cummins*[1], *Tomas Gedeon*[1]*, *Shaun Harker*[2] *and Konstantin Mischaikow*[2]

[1] *Department of Mathematical Sciences, Montana State University, Bozeman, MT, United States,* [2] *Department of Mathematics, Rutgers, The State University of New Jersey, New Brunswick, NJ, United States*

We present a computational tool DSGRN for exploring the dynamics of a network by computing summaries of the dynamics of switching models compatible with the network across all parameters. The network can arise directly from a biological problem, or indirectly as the interaction graph of a Boolean model. This tool computes a finite decomposition of parameter space such that for each region, the state transition graph that describes the coarse dynamical behavior of a network is the same. Each of these parameter regions corresponds to a different logical description of the network dynamics. The comparison of dynamics across parameters with experimental data allows the rejection of parameter regimes or entire networks as viable models for representing the underlying regulatory mechanisms. This in turn allows a search through the space of perturbations of a given network for networks that robustly fit the data. These are the first steps toward discovering a network that optimally matches the observed dynamics by searching through the space of networks.

**Keywords: Boolean networks, switching systems, network dynamics, parameter space, database of dynamics**

OPEN ACCESS

***Edited by:***
*Matteo Barberis,*
*University of Amsterdam, Netherlands*

***Reviewed by:***
*Tomáš Helikar,*
*University of Nebraska-Lincoln, United States*
*Kyle B. Gustafson,*
*United States Department of the Navy, United States*
*Marija Cvijovic,*
*Chalmers University of Technology, Sweden*

***Correspondence:**
*Tomas Gedeon*
*gedeon@math.montana.edu*

***Specialty section:***
*This article was submitted to Systems Biology, a section of the journal Frontiers in Physiology*

***Received:*** *31 January 2018*
***Accepted:*** *30 April 2018*
***Published:*** *23 May 2018*

***Citation:***
*Cummins B, Gedeon T, Harker S and Mischaikow K (2018) DSGRN: Examining the Dynamics of Families of Logical Models. Front. Physiol. 9:549. doi: 10.3389/fphys.2018.00549*

## 1. INTRODUCTION

Experimentally determined pairwise interactions between genes, proteins and signaling molecules are often assembled into pathways and networks. There is a strong desire to understand the dynamics of networks, diversity of their potential stable behavior, as well their response under mutations or targeted pharmacological intervention. Such an ability would allow us to target many diseases, most importantly cancer, with great precision and accuracy, without disturbing other functions of the cell, and without the devastating side effects on healthy cells that are the hallmark of many current drugs.

The current state of modeling gene network dynamics is characterized by a trade-off between the model's ability to quantitatively match the experimental data, and the need for a large number of kinetic parameters to parameterize the model (Karlebach and Shamir, 2008; Heatha and Kavria, 2009; Machado et al., 2011; Goncalves et al., 2013). Properly parameterized ordinary differential equation models can provide a good quantitative match and are easily generalized (Chen et al., 2004; Tyson and Novak, 2013). However, numerical simulation of these models require knowledge of kinetic parameters that are usually not known. The indirect estimate of these parameters by comparing the output of the model to the experimental data suffers from at least three fundamental problems: (i) the correspondence between dynamics and the structure of the network is not one-to-one; (ii) the need to match data corrupted by significant intrinsic and experimental noise to an individual solution of the ODE model; and (iii) the lack of methods to search high dimensional parameter spaces for dynamic signatures observed in the data.

A popular modeling platform is that of Boolean nets, where each protein, ligand or mRNA is assumed to have two states (ON and OFF), and the discrete time evolution of the states is based on logic-like update functions (Glass and Kauffman, 1972, 1973; Thomas, 1973; Thomas et al., 1995; von Dassow et al., 2000; Bernard and Gouze, 2002; de Jong, 2002; de Jong et al., 2004; Belta and Habets, 2006; Chaves et al., 2006; Faure et al., 2006; Albert, 2007; Batt et al., 2007a,b; Bornholt, 2008; Tournier and Chaves, 2009; Machado et al., 2011; Albert et al., 2013; Saadatpour and Reka, 2013). Rather than providing rate parameters, the biological input into model formulation is limited to postulating logical functions, one for each node in the network, which compute the next Boolean state of node $i$ based on Boolean states of the nodes that provide input to node $i$. These Boolean functions at each node are assembled into a Boolean function that provides the next state of all nodes in the network based on the previous state of the network. Iterations of this function are an approximation of the time evolution of the state of the network.

This attractive class of synchronous Boolean models has several disadvantages. The first class of objections comes from biology: these models cannot represent ubiquitous cellular noise, since states change simultaneously they require unreasonable uniformity of duration of different cellular processes, and the fit to experimental data is typically problematic. A mathematical objection is that discretization of the phase space and the discretization of the set of Boolean functions compatible with a given network does not allow consideration of changing dynamics under graded perturbation. In other words, it is difficult to construct a bifurcation theory in the class of Boolean functions.

In this contribution we study multi-level discrete maps, which are a direct generalization of Boolean maps, that are compatible with an ODE system. We propose that only the asynchronous updates of these discrete maps have biological meaning. The concept of an asynchronous update of a Boolean function has been introduced previously (Pauleve and Richard, 2012). We review and formalize these concepts in the next section. We then study a particular class of ODEs that can be viewed as a continuous parameterization of a family of multi-level discrete maps. Continuous parameterization of a finite number of inherently discrete objects implies that there is a finite decomposition of the parameter space into disjoint domains, each of which supports a multi-level discrete map. Mutual position of these parameter domains is captured in a *parameter graph*, whose nodes represent the domains and edges their adjacency.

We describe a computational approach, called Dynamic Signatures Generated by Regulatory Networks (DSGRN), that computes the parameter graph for a given network and input interaction at each node. In addition, to each node of the parameter graph we associate a *Morse graph* whose nodes are the strongly connected path components of the asynchronous update of the corresponding multi-level discrete map, and edges represent reachability by iterations of this map. We call the resulting collection a *DSGRN Database*.

## 2. BASIC DEFINITIONS

**Definition 2.1.** A *regulatory network* $\mathbf{RN} = (V, E)$ is a graph with network nodes $V = \{1, 2, \ldots, N\}$ and signed, directed edges $E \subset V \times V \times \{\rightarrow, \dashv\}$. For $i, j \in V$, we will use the notation $(i, j) \in E$ to denote a directed edge from $i$ to $j$ of either sign, $i \rightarrow j$ to denote an *activation* or positive interaction, and $i \dashv j$ to denote a *repression* or negative interaction.

We define the *targets* of a node $i$ as

$$\mathbf{T}(i) := \{j \mid (i, j) \in E\}$$

and the *sources* of a node $i$ as

$$\mathbf{S}(i) := \{j \mid (j, i) \in E\}$$

For each node $i$ in a regulatory network $\mathbf{RN}$, define a set of *integer states* $\mathcal{V}(i) := \{0, 1, \ldots, m_i\}$. Let $\mathcal{V} := \prod_{i=1}^{N} \mathcal{V}(i)$. For state $s \in \mathcal{V}$ let

$$S_{+}^{i} := \{u \in \mathcal{V} \mid u_i > s_i, u_j = s_j \text{ for all } j \neq i\}$$

be the set of states that differ from $s$ only in the $i$-th coordinate and are strictly greater in the $i$-th coordinate.

**Definition 2.2.** We say a (multi-valued) map $f : \mathcal{V} \rightarrow \mathcal{V}$ is *compatible* with a regulatory network $\mathbf{RN}$ ($\mathbf{RN}$-compatible) if and only if the following are satisfied

- $(i, j) \in E$ is a positive edge $i \rightarrow j$ if and only if there exists a state $s \in \mathcal{V}$ and at least one $u \in S_{+}^{i}$ such that $f_j(u) > f_j(s)$.
- $(i, j) \in E$ is a negative edge $i \dashv j$ if and only if there exists a state $s \in \mathcal{V}$ and at least one $u \in S_{+}^{i}$ such that $f_j(u) < f_j(s)$.

A regulatory network, as introduced in this paper, is also called the *interaction graph* of Boolean function $f$, as defined in Pauleve and Richard (2012). Our definition above goes in the opposite direction and defines a set of multivalued maps consistent with a fixed regulatory network; we also generalize from Boolean maps to maps with more than two discrete values.

**Definition 2.3.** A *synchronous Boolean model for a regulatory network* ***RN*** is an $\mathbf{RN}$-compatible map

$$B : \{0, 1\}^N \rightarrow \{0, 1\}^N.$$

Given a synchronous Boolean model $B$, the regulatory network $\mathbf{RN}$ such that $B$ is $\mathbf{RN}$-compatible, is the *interaction graph of* $B$.

**Definition 2.4.** A *synchronous multi-level discrete map for a regulatory network* ***RN*** is an $\mathbf{RN}$-compatible map

$$D : \mathcal{V} \rightarrow \mathcal{V}$$

where $\mathcal{V} = \prod_{i=1}^{N} \{0, 1, \ldots, m_i\}$.

**Definition 2.5.** A *nearest neighbor multi-valued map for a regulatory network* ***RN*** is an $\mathbf{RN}$-compatible map

$$\mathcal{F} : \mathcal{V} \rightrightarrows \mathcal{V}$$

such that either $s \in \mathcal{F}(s)$ or, if $v \in \mathcal{F}(s)$ and $v \neq s$ then $v$ satisfies the *adjacency condition*:

$$|v_i - s_i| = \begin{cases} 1, \text{ for } i = k \\ 0, \text{ for } i \neq k \end{cases}$$

for exactly one index $k$. We say that $s$ and $v$ are *adjacent.*

**Definition 2.6.** We say a nearest neighbor multi-valued map $\mathcal{F}$ is *an asynchronous update* of a multi-level discrete map $D$ if, given

$$t_1 = D(s_1) \qquad \text{where} \qquad t_1 = (t_{1,1}, \ldots, t_{1,N}) \text{ and } s_1 = (s_{1,1}, \ldots, s_{1,N}),$$

we have $s_2 \in \mathcal{F}(s_1)$ in either of the two following conditions:

(a) if $t_1 = s_1$ then $s_2 = s_1$; or
(b) if $t_1 \neq s_1$, then $s_2$ is adjacent to $s_1$, and $s_2$ lies between $s_1$ and $t_1$, which means that either
 (a) $s_{1,i} < s_{2,i} \leq t_{1,i}$ or
 (b) $s_{1,i} > s_{2,i} \geq t_{1,i}$.

For a regulatory network $\mathbf{RN} = (V, E)$ consider a system of ODEs in variables $x_i$ for each $i \in V$. We assume that there are finite number of thresholds $\theta_{1,i}, \ldots, \theta_{m_i,i}$ that divide the semi-axis $[0, \infty)$ to $m_i + 1$ intervals $I_k$. The collection of thresholds $\{\theta_{j,i}\}$ decomposes $[0, \infty)^N$ into a finite number of domains $\kappa$, characterized by the property that the projection on $i$-th variable $\pi_i(\kappa) = I_k$ for a unique $k \in \{0, \ldots, m_i\}$ for every $i$. We call each $\kappa$ a *domain*. Let $\mathcal{K}$ be a collection of all domains $\kappa \subset \mathbb{R}^{N+}$ in the non-negative orthant of $\mathbb{R}^N$.

Let $x = (x_1, \ldots, x_N) \in \mathbb{R}^{N+}$ and let

$$G_i : [0, \infty) \to \mathcal{V}(i)$$

be defined by $G_i(x_i) = k$ if and only if $x_i \in I_k$. Let

$$G : [0, \infty)^N \to \mathcal{V}$$

be the vector-valued function with coordinate functions $G_i$. For a given domain $\kappa$, the value $G(x)$ does not depend on $x \in \kappa$. Therefore we can assign the *state* $s := G(x) \in \mathcal{V}, x \in \kappa$ to the domain $\kappa$ and write $s = g(\kappa)$. Viewed as a map on the set of domains $\mathcal{K}, g$ is a bijection

$$g : \mathcal{K} \to \mathcal{V}.$$

**Definition 2.7.** For a regulatory network $\mathbf{RN} = (V, E)$ consider a system of ODEs in variables $x_i$ for each $i \in V$. We say that such an ODE system is *compatible* with a nearest neighbor multi-valued map $\mathcal{F}$ if solutions $x(t)$ can traverse from domain $\kappa_1$ to adjacent domain $\kappa_2$ only if $g(\kappa_2) \in \mathcal{F} \circ g(\kappa_1)$.

This definition of compatible ODE system states that the dynamics of an ODE system can be captured, in an coarse sense, by a finite multi-valued map. We now apply these ideas to a specific family of ODE systems.

## 3. SWITCHING SYSTEMS

Switching systems, also known as Glass systems, were introduced by Glass (Glass and Kauffman, 1972, 1973) in the 1970's and developed subsequently by many authors (Thomas, 1973; Thomas et al., 1995; Edwards, 2001; Bernard and Gouze, 2002; de Jong, 2002; de Jong et al., 2004; Chaves et al., 2006; Tournier and Chaves, 2009; Ironi et al., 2011; Edwards et al., 2015).

**Definition 3.1.** A *switching system* for a regulatory network $\mathbf{RN}= (V, E)$ is a system of ordinary differential equations

$$\dot{x}_i = -\gamma_i x_i + M_i \circ \sigma_i(x), i \in V \tag{1}$$

where $\gamma_i > 0$ is a decay rate, $M_i$ is a multi-affine algebraic expression (Belta and Habets, 2006; Batt et al., 2007b; Cummins et al., 2016) , and $\sigma_i = (\sigma_{i,j})$ is a vector of step functions, one for each edge $(j, i) \in E$. When $(j, i) = j \to i$ is an activation, then the step function transitions from a low $(l_{i,j})$ to a high value $(u_{i,j})$, and when $(j, i) = j \dashv i$ is a repression, then $\sigma_{i,j}$ transitions from $u_{i,j}$ to $l_{i,j}$. The transition happens at the threshold $x_j = \theta_{j,i}$:

$$\sigma_{i,j} := \begin{cases} l_{i,j} & \text{if } j \to i \in E \text{ and } x_j < \theta_{i,j} \\ & \text{or } j \dashv i \in E \text{ and } x_j > \theta_{i,j} \\ u_{i,j} & \text{if } j \to i \in E \text{ and } x_j > \theta_{i,j} \\ & \text{or } j \dashv i \in E \text{ and } x_j < \theta_{i,j} \end{cases} \tag{2}$$

We assume $0 < \theta_{i,j}$ and $0 < l_{i,j} < u_{i,j}$ to ensure the model captures the basic biological meaning of concentration, activation, and repression. We further assume $\theta_{i,j} \neq \theta_{k,j}$ for all $j \in V$ whenever $i \neq k$ and so each node $j$ affects its downstream nodes at different thresholds.

It is important to note that to a given $\mathbf{RN}$ one can associate many switching systems. Indeed, a selection of multi-linear expressions $M_i, i = 1, \ldots, N$ in addition to the structure of the network $\mathbf{RN}$, determines the parameterized set of ODEs (1). The function $M_i$ determines how the information from the source nodes $\mathbf{S}(i)$ is combined into the right hand side of (1).

A *parameter* of the switching system is a set of real numbers

$$p = \{\gamma_i \mid i \in V\} \cup \{\theta_{i,j}, l_{i,j}, u_{i,j} \mid (j, i) \in E\}$$

that satisfy these constraints. The set of all parameters $p$ is denoted $P$.

**Definition 3.2.** The collection $\Theta_i : = \{\theta_{j,i} \mid j \in \mathbf{T}(i)\}$ for each node $i \in V$ is totally ordered, and this order induces a decomposition of phase space $\mathcal{K}$, such that each domain $\kappa \in \mathcal{K}$ is written

$$\kappa = \prod_i [\theta_{j_k,i}, \theta_{j_{k+1},i}]$$

where $\theta_{j_k,i}, \theta_{j_{k+1},i}$ are adjacent. We define the thresholds $\theta_{0,i} : = 0$ and $\theta_{\infty,i} : = \infty$, so that the intervals below the lowest threshold and above the highest threshold are captured.

Let $m_i = |\mathbf{T}(i)|$ be the number of targets of node $i \in V$, and let $\mathcal{V} = \prod_{i=1}^{N}\{0, 1, \ldots, m_i\}$ as before. The decomposition $\mathcal{K}$ is the same as that in the previous section, and using the total order on $\Theta_i$, we can construct an appropriate bijection $g : \mathcal{K} \to \mathcal{V}$. Using this bijection $g$, we show in Crawford-Kahrl et al. (2018) that given a switching system at a fixed parameter $p \in P$, there is a unique multi-level discrete map $D^p$, and an asynchronous update rule of $D^p$, $\mathcal{F}^p$, such that the switching system is compatible with $\mathcal{F}^p$. We note that the collection $\{D^p\}_{p \in P}$ does not exhaust the entire collection of **RN**-compatible multi-level maps $D$. However, the induced collection of maps $\{D^p\}_{p \in P}$ decomposes into finite number of classes.

**Definition 3.3.** Let $p$ be a parameter of a switching system with totally ordered thresholds $\Theta_i^p$. Let $D^p$ be the unique multi-level function associated to the switching system parameterized by $p$. Let $O_i^p = \{j_1 < j_2 < \cdots < j_{m_i}\}$ be such that $j_k < j_l$ if and only if $\theta_{j_k,i} < \theta_{j_l,i}$ in $\Theta_i^p$. Define $O^p = \{O_i^p\}$ to be the *order parameter* associated to $p$, and $(O^p, D^p)$ to be the *combinatorial parameter* of the system. If $q$ is another parameter of the switching system with $(O^q, D^q)$, then we define an equivalence relation $q \sim p$ when $(O^q, D^q) = (O^p, D^p)$. We call the collection of combinatorial parameters $\mathcal{Z}$.

The partition induced by $\sim$ is clearly finite, since the order of $m_i$ integers is finite, and the number of multi-level maps $D$ on a finite set is also finite. Let $s := |\mathcal{Z}|$ be the cardinality of the set $\mathcal{Z}$. We show in Cummins et al. (2016) that each $u \in \mathcal{Z}$ has a computable geometrical representation as a connected subset $U \subset P$. Therefore there is a computable decomposition of the parameter space $P$ in $s$ regions $U_i$ for $i = 1, \ldots, s$, such that for any $p, q \in U_i$ we have $D^p = D^q$, and hence also $\mathcal{F}^p = \mathcal{F}^q$. Therefore a finite collection $\{\mathcal{F}^u\}_{u \in \mathcal{Z}}$ captures dynamics of all maps $\mathcal{F}^p$ across all the parameter space $P$.

We remark that the parameter graph captures the dynamics of all subgraphs of **RN** as well as **RN** itself. Although not addressed in this paper, we can limit the exploration of the dynamics only to those combinatorial parameters that result in **RN**-compatible multi-level discrete maps $D$.

## 4. DSGRN: DYNAMICAL SIGNATURES GENERATED BY REGULATORY NETWORKS

Given a network **RN** and the associated switching system, the computational tool DSGRN (Cummins et al., 2016; Harker, 2018) computes and records a graph of graphs in SQL database format. This general database can be queried in many ways, and we will give a short example after defining the graphs that are computed. If a user starts with a synchronous Boolean model $B$, the first step is to calculate an the interaction graph **RN** of $B$. DSGRN then describes the long term dynamics of all multi-valued nearest neighbor maps compatible with the switching systems associated to **RN**. Each of these multi-valued nearest neighbor maps is an asynchronous update of a multi-level discrete map. Therefore DSGRN embeds the dynamics of $B$ into a family of multi-level discrete models that are all compatible with the dynamics of a switching system associated to **RN**.

**Definition 4.1.** The parameter graph $\mathcal{P} = (C, A)$ has nodes $C$ that represent all combinatorial parameters via a bijection $h : C \to \mathcal{Z}$. The non-directed edges $(c, c') \in A$ occur when the difference between $h(c) = (O, D)$ and $h(c') = (O', D')$ is exactly one of the following:

1. there is a swap in the order of one pair of adjacent integers $j_k, j_l$ between $O$ and $O'$, and all other elements remain the same;
2. for exactly one $v \in \mathcal{V}$, $||D(v) - D(v')|| = 1$, and $||D(w) - D'(w)|| = 0$ for all $w \in \mathcal{V} \setminus \{v\}$.

For each $u \in \mathcal{Z}$, there is a representative nearest-neighbor multi-valued discrete map $\mathcal{F}^u$. This map can be viewed as a graph.

**Definition 4.2.** The *state transition graph (STG)* of a switching system with combinatorial parameter $u$ is the directed graph $(\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V}$ were defined previously, and $(v, w) \in \mathcal{E}$ if and only if $w \in \mathcal{F}^u(v)$.

A *recurrent component* (also referred to as a *strongly connected path component*) of the STG $(\mathcal{V}, \mathcal{E})$ is a maximal collection $\mathcal{M}$ of vertices such that for any $u, v \in \mathcal{M}$ there exists a non-empty path from $u$ to $v$ within the subgraph induced by $\mathcal{M}$. The collection of all recurrent components of $(\mathcal{V}, \mathcal{E})$ is denoted by

$$\mathsf{MD}(\mathcal{F}) := \{\mathcal{M}(i) \subset \mathcal{V} \mid i \in \mathsf{P}\}$$

and is called a *Morse decomposition* of the STG. Here $\mathsf{P}$ is an index set. Recurrent components inherit a well-defined partial order by the reachability relation in the directed graph $(\mathcal{V}, \mathcal{E})$. In particular, there is a partial order on the indexing set $\mathsf{P}$ of $\mathsf{MD}(\mathcal{F})$ defined by i ≤ j if there exists a path in $(\mathcal{V}, \mathcal{E})$ from an element of $\mathcal{M}(j)$ to an element of $\mathcal{M}(i)$.

**Definition 4.3.** The *Morse graph* of the STG, denoted $\mathsf{MG}(\mathcal{F})$, is the Hasse diagram of the poset $(\mathsf{P}, \leq)$. We refer to the elements of $\mathsf{P}$ as the *Morse nodes* of the graph.

Any recurrent behavior of the ODE system will be be captured by one of the Morse nodes of the Morse graph. That is, any recurrent set of the ODE will be a subset of a set of domains that correspond to states in STG that belong to a single Morse node.

Each component of the Morse graph can be annotated. We use the following terminology:

1. FP denotes a Morse graph component consisting of a single node of the state transition graph (STG).
2. FP$(v)$ denotes an FP that is located in $\kappa = g^{-1}(v)$ for $v \in \mathcal{V}$.
3. FP ON denotes an FP in which the associated $v$ has no zeros.
4. FP OFF denotes an FP in which the associated $v$ is all zeros.
5. FC denotes a Morse graph component $\mathcal{M}$ that contains at least one path through the subgraph induced by $\mathcal{M}$ that crosses at least one threshold in each variable $x_i$. FC stands for "full cycle."
6. XC$(x_{j_1}, \ldots, x_{j_n})$ denotes a partial oscillation in variables $x_{j_1}, \ldots, x_{j_n}$, where only thresholds in these variables are crossed by paths in the Morse graph component.

If a component is a leaf of the Morse graph, i.e., it has no outgoing edges, then we call it an *attractor*. For each node in the parameter graph, DSGRN records the annotated Morse graph, and this collection comprises the database.

## 5. EXAMPLE

A DSGRN Database can queried via any general expression in SQL. Some queries have been implemented on a sample set of databases at http://chomp.rutgers.edu/Projects/DSGRN/DB/index.html. See **Figure 1A** for a screenshot of the above website showing networks with precomputed databases. This screenshot shows a selection of different regulatory networks, each of which may be clicked on to show detailed information about the computation of the network dynamics. **Figure 1B** shows a screenshot the result of such a click, and **Figure 1B** shows the result of applying a filter to the network dynamics. We will now step through each of these screenshots in more detail to explain the displayed summary of network dynamics.

In **Figure 1A**, in the third row on the right, we see a network labeled `5D_2015_10_21_VA`. Clicking on it, we see the middle screenshot in **Figure 1B**. The picture of the network **RN** is in the upper left, and next to it an Annotation Filter, which allows us to filter the results based on the annotations of the displayed Morse graphs. All of the annotated Morse graphs that are generated by at least one combinatorial parameter are shown, ordered by the number of combinatorial parameters that produced the given Morse graph. By clicking on the “Yes” button beside FC, we select the Morse graphs that contain a component annotated by FC. In **Figure 1C**, we show a few top Morse graphs satisfying this condition. By choosing different combinations of “Yes”, “No”, and “Either” in the Annotation Filter, we can explore the different dynamical behaviors of the system.

Although graphical display of the database is useful for exploratory purposes, it is not as powerful as SQL searches over the DSGRN database in which arbitrary combinations of annotated Morse graphs can be selected. Moreover, to use graphical display it is necessary to set up a server. The expected use of DSGRN is to calculate the database and then to use flexible, user-defined SQL queries to search for dynamics of interest.

We now show how to perform some queries that are not available in our demo website. In order to compute the database for DSGRN, the user needs to install DSGRN (Harker, 2018) from GitHub, following the instructions on http://dsgrn.readthedocs.io/en/latest/index.html. While we intend to provide SBML compatibility in the near future, currently the user needs to create a network file that provides names for each node in the regulatory network **RN** and describes the input logic function $M_i$ for each node $i$. The following is the network file for `5D_2015_10_21_VA` as shown in the upper left of the middle screenshot in **Figure 1B**:

```
p53 : (Chk2 + ATM)(∼Mdm2)
ATM : ∼Wip1
Chk2 : ATM (∼Wip1)
Wip1 : p53
Mdm2 : p53
```

The name of the node is on the left hand side of the colon, and the input logic function $M_i$ to the node is on the right hand side. For example, p53 has three inputs, with “OR” (addition) logic between Chk2 and ATM, and “AND NOT” (multiplication) logic on Mdm2. The symbol “∼” denotes repression. Suppose that this file is saved under “RN.txt.” To compute the DSGRN SQL Database named “RN.db” using 4 threads we run the following command:

```
  mpiexec -np 4 Signatures RN.txt RN.db
```

After the database is computed, we can query RN.db for different dynamical behaviors. Several tables for the database are automatically generated, including Signatures, MorseGraphAnnotations, and MorseGraphEdges, which we will use in queries below. For a comprehensive list of the tables generated, more detail on the SQL database, and other queries, see the links from the documentation site http://dsgrn.readthedocs.io/en/latest/index.html.

We take the number of combinatorial parameters that generates a specific dynamical behavior to be a proxy for the robustness of the behavior across all of parameter space. The number of combinatorial parameters for network **RN** specified in RN.txt is the number of rows in the database RN.db. Therefore we can find the number of parameters using the command:

```
  sqlite3 RN.db ‘select count(*) from
Signatures’
```

which in this case tells us that there are 803,520 parameters associated to the network `5D_2015_10_21_VA`. We now search the database for the number of combinatorial parameters with at least one *stable* FC. Note that the Annotation Filter in **Figure 1B** searches for any FC, including unstable ones. The command for this search is

```
  sqlite3 RN.db ‘select count(*) from
Signatures natural join
(select distinct(MorseGraphIndex) from
(select MorseGraphIndex,Vertex from
MorseGraphAnnotations where Label="FC"
except select MorseGraphIndex,Source from
MorseGraphEdges))’
```

and the result is 6904 combinatorial parameters, which is 0.86% of all the parameters. In contrast, the number with at least one stable FP is 667,536, which is 83% of the parameters, obtained by:

```
  sqlite3 RN.db ‘select count(*) from
Signatures natural join
(select distinct(MorseGraphIndex) from
(select MorseGraphIndex,Vertex from
MorseGraphAnnotations where Label like
"FP%"
except select MorseGraphIndex,Source from
MorseGraphEdges))’
```

Based on the results of these queries, we conclude that a stable FP is far more common that a stable FC, and therefore a more robust behavior for this network.

**Table 1** shows the computational scaling of DSGRN in a series of small networks taken from http://chomp.rutgers.edu/Projects/DSGRN/DB/index.html, some of which are shown **Figure 1A**. We see that the computation time and database storage increase

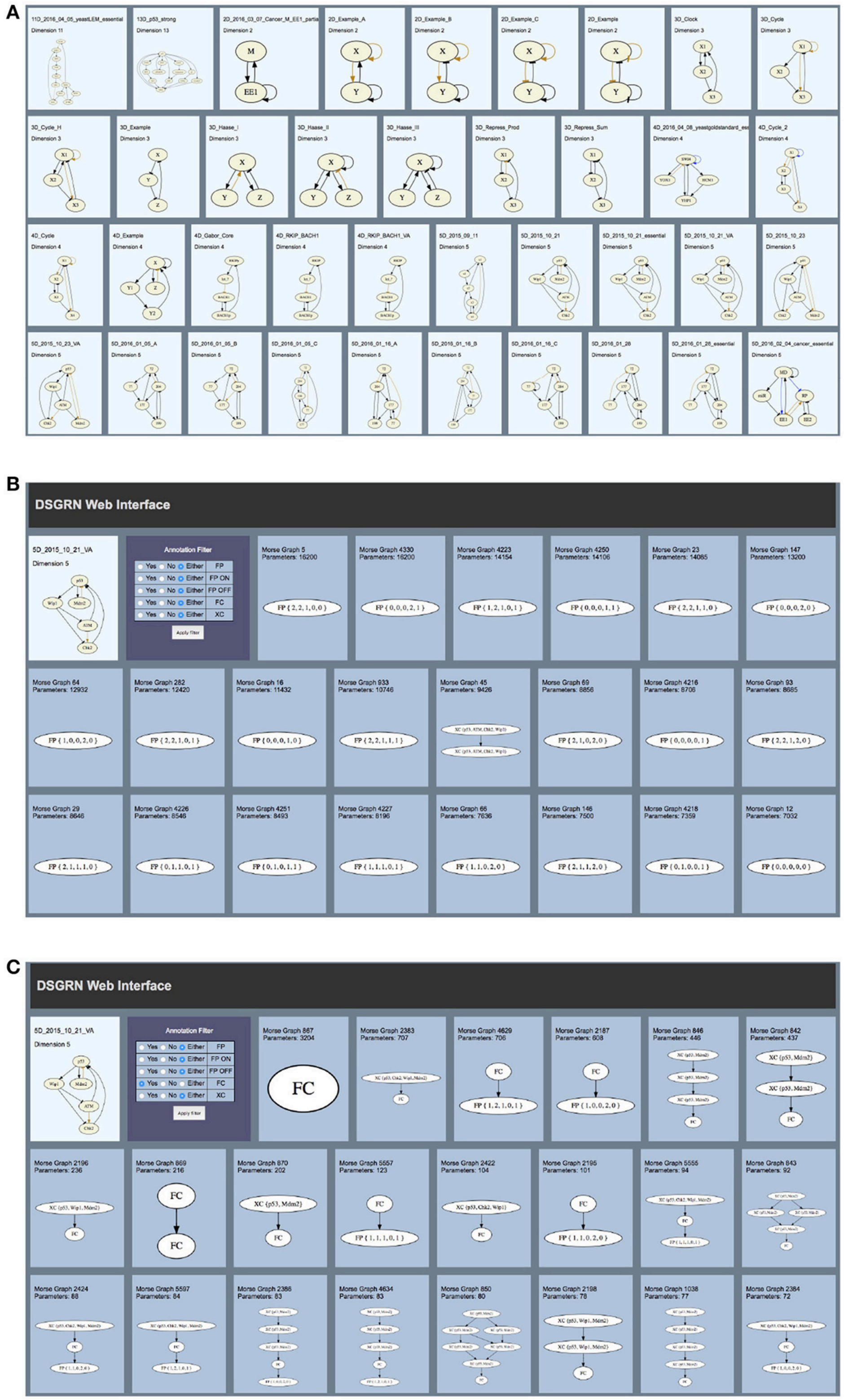
A
B
DSGRN Web Interface
5D_2015_10_21_VA
Dimension 5
Annotation Filter
Yes No Either FP
Yes No Either FP ON
Yes No Either FP OFF
Yes No Either FC
Yes No Either XC
Apply filter
Morse Graph 5 Parameters: 16200
Morse Graph 4330 Parameters: 16200
Morse Graph 4223 Parameters: 14154
Morse Graph 4250 Parameters: 14106
Morse Graph 23 Parameters: 14085
Morse Graph 147 Parameters: 13200
Morse Graph 64 Parameters: 12932
Morse Graph 282 Parameters: 12420
Morse Graph 16 Parameters: 11432
Morse Graph 933 Parameters: 10746
Morse Graph 45 Parameters: 9426
Morse Graph 69 Parameters: 8856
Morse Graph 4216 Parameters: 8706
Morse Graph 93 Parameters: 8685
Morse Graph 29 Parameters: 8646
Morse Graph 4226 Parameters: 8546
Morse Graph 4251 Parameters: 8493
Morse Graph 4227 Parameters: 8196
Morse Graph 66 Parameters: 7636
Morse Graph 146 Parameters: 7500
Morse Graph 4218 Parameters: 7359
Morse Graph 12 Parameters: 7032
C
DSGRN Web Interface
Morse Graph 867 Parameters: 3204
Morse Graph 2383 Parameters: 707
Morse Graph 4629 Parameters: 706
Morse Graph 2187 Parameters: 608
Morse Graph 846 Parameters: 446
Morse Graph 842 Parameters: 437
Morse Graph 2196 Parameters: 236
Morse Graph 869 Parameters: 216
Morse Graph 870 Parameters: 202
Morse Graph 5557 Parameters: 123
Morse Graph 2422 Parameters: 104
Morse Graph 2195 Parameters: 101
Morse Graph 5555 Parameters: 94
Morse Graph 843 Parameters: 92
Morse Graph 2424 Parameters: 88
Morse Graph 5597 Parameters: 84
Morse Graph 2386 Parameters: 83
Morse Graph 4634 Parameters: 83
Morse Graph 850 Parameters: 80
Morse Graph 2198 Parameters: 78
Morse Graph 1038 Parameters: 77
Morse Graph 2384 Parameters: 72
FC
XC (p53, Mdm2)
XC (p53, Wip1, Mdm2)

**FIGURE 1 |** Screenshots of http://chomp.rutgers.edu/Projects/DSGRN/DB/index.html. The description of the Figure and step-by-step guide through an example is in the text.

**TABLE 1 |** Example performance of DSGRN on 4 threads on a 2013 MacBook Pro. In practice, DSGRN is limited more by storage space than by computation time.

| Name | # Nodes | # Edges | # Parameters | Time | Storage |
|---|---|---|---|---|---|
| `2D_Example_A` | 2 | 4 | 1,600 | 2.7 s | 124 K |
| `3D_Cycle` | 3 | 5 | 5,400 | 3.1 s | 224 K |
| `4D_Example` | 4 | 6 | 122,472 | 10.4 s | 4 M |
| `5D_2015_10_21_VA` | 5 | 8 | 803,520 | 2 m 26 s | 46 M |
| `7D_2016_04_05_yeastLEM` | 7 | 10 | 3,499,200 | 12 m 41 s | 128 M |

rapidly as the network size increases. This increase is due particularly to the presence of high degree nodes, rather than to the absolute number of nodes and edges. High degree nodes cause the most rapid increase in the number of combinatorial parameters. Because of parallelization and usage of computing clusters with a large core count, we find in practice that DSGRN is more limited by space to store databases than by computation time.

In order to address the storage space scaling limitations, we have implemented two additions to DSGRN. The first is the idea of "essential" parameters, which is the subset of parameters consistent with Definition 2.2. DSGRN was originally designed to study not only **RN**-compatible asynchronous multi-level maps, but all such maps that were **S**-compatible with any subgraph **S** of **RN**. By limiting ourselves to **RN**-compatible maps, the size of parameter space is greatly reduced. To specify essential parameters, add "`: E`" to the end of every line in the network specification file for **RN**. For example, the essential network specification file for `2D_Example_A` using multiplicative logic is:

```
X : XY : E
Y : XY : E
```

The second addition is an extensive Python module DSGRN that can be used to explore individual parameters rather than calculating the entire database at once. This model is part of the standard DSGRN installation. If a hypothesis about the network dynamics can be constructed a priori, then the selection for annotated Morse graphs can be computed on the fly, allowing much larger networks to be analyzed than is otherwise possible. See https://github.com/shaunharker/DSGRN/blob/master/Tutorials/GettingStarted.ipynb for a brief introduction to the Python library.

## 6. DISCUSSION

Given a regulatory network **RN** there is a very large number of multi-level maps $D$ that can be associated to this network. We can enumerate them by selecting for each node an arbitrary assignment of node value based on the node inputs. If the structure of the network is the only information available, these all represent valid models for the network dynamics in the class of discrete multi-level maps, which generalize Boolean models. This class of functions generate, via asynchronous update, a class of multi-valued nearest neighbor maps $\mathcal{F}$ which better represent biological reality. States of $\mathcal{F}$ only change one at a time.

To make the collection of **RN**-compatible functions $\mathcal{F}$ smaller and more biologically realistic, we employ a switching system, which is an ODE system with discrete-valued interaction terms. They were introduced in the 1970's (Glass and Kauffman, 1972, 1973) as a continuous time counterpart to Boolean networks. A switching system is parameterized by continuous parameters, but this set decomposes into a finite number of computable regions (Cummins et al., 2016), each of which is associated with a single multi-level map $D^u$ and its asynchronous update $\mathcal{F}^u$, where $\mathcal{F}^u$ is compatible with the switching system ODE (Crawford-Kahrl et al., 2018). The mutual position of these regions in the parameter space provide a natural way to define a notion of "neighboring" functions $D^u, D^v$ (and thus $\mathcal{F}^u, \mathcal{F}^v$).

Our computational tool DSGRN (Cummins et al., 2016; Cummins et al., 2017; Harker, 2018) constructs the collection of all such parameter regions and encodes them in the form of a parameter graph. For each node $u$ of the parameter graph, the DSGRN Database stores information about the global dynamics in form of a Morse graph, which is a summary of the dynamics of $\mathcal{F}^u$. A DSGRN Database provides a summary of dynamics for all maps $\mathcal{F}^u$ which are compatible with a switching system on **RN**. In this sense DSGRN represents the dynamics compatible with the network **RN** across all parameters.

DSGRN can be used to either list dynamical behaviors that are compatible with a given network **RN**, or search in the space of networks for those networks that provide most robustly dynamics of interest, for instance FC or FP.

## AUTHOR CONTRIBUTIONS

TG, KM conceptualized the paper. TG, BC wrote the paper. SH, BC implemented the methods and performed computations.

## ACKNOWLEDGMENTS

TG was partially supported by NSF grants DMS-1226213, DMS-1361240, USDA 2015-51106-23970, DARPA grants D12AP200025 and FA8750-17-C-0054, and NIH grants 1R01AG040020-01 and 1R01GM126555-01. BC was partially supported by grants USDA 2015-51106-23970, DARPA grants D12AP200025 and FA8750-17-C-0054 and NIH 1R01GM126555-01. The work of SH and KM was partially supported by grants NSF-DMS-1125174, 1248071, 1521771, NIH 1R01GM126555-01 and DARPA contracts HR0011-16-2-0033, FA8750-17-C-0054. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

## REFERENCES

Albert, R. (2007). Network inference, analysis, and modeling in systems biology. *Plant Cell* 19, 3327–3338. doi: 10.1105/tpc.107.054700

Albert, R., Collins, J. J., and Glass, L. (2013). Introduction to focus issue: quantitative approaches to genetic networks. *Chaos* 23:025001. doi: 10.1063/1.4810923

Batt, G., Belta, C., and Weiss, R. (2007a). "Model checking genetic regulatory networks with parameter uncertainty," in *Hybrid Systems: Computation and Control, HSCCÕ07, Lecture Notes in Computer Science 4416* (Berlin: Springer), 61–75.

Batt, G., Yordanov, B., Weiss, R., and Belta, C. (2007b). Robustness analysis and tuning of synthetic gene networks. *Bioinformatics* 23, 2415–2422. doi: 10.1093/bioinformatics/btm362

Belta, C., and Habets, L. (2006). Controlling a class of nonlinear systems on rectangles. *Trabs. Aut. Control* 51, 1749–1759. doi: 10.1109/TAC.2006.884957

Bernard, O., and Gouze, J. (2002). Global qualitative description of a class of nonlinear dynamical systems. *Artif. Intell.* 136, 29–59. doi: 10.1016/S0004-3702(01)00169-2

Bornholt, S. (2008). Boolean network models of cellular regulation: prospects and limitations. *J. R. Soc. Interface* 5, 134–150. doi: 10.1098/rsif.2008.0132.focus

Chaves, M., Sontag, E. D., and Albert, R. (2006). Methods of robustness analysis for Boolean models of gene control networks. *IEE Proc. Syst. Biol.* 153, 154–167. doi: 10.1049/ip-syb:20050079

Chen, K., Calzone, L., Csikasz-Nagy, A., Cross, F., Novak, B., and Tyson, J. (2004). Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell* 15, 3841–3862. doi: 10.1091/mbc.e03-11-0794

Crawford-Kahrl, P., Cummins, B., and Gedeon, T. (2018). Comparison of two combinatorial models of global network dynamics. arXiv 1801.06524.

Cummins, B., Gedeon, T., Harker, S., and Mischaikow, K. (2017). "Database of dynamic signatures generated by regulatory networks (DSGRN)," in *Computational Methods in Systems Biology - 2017*, eds J. Feret and H. Koeppl (Cham: Springer), 300–308. Available online at: https://www.springer.com/us/book/9783319674704

Cummins, B., Gedeon, T., Harker, S., Mischaikow, K., and Mok, K. (2016). Combinatorial representation of parameter space for switching systems. *SIAM J. Appl. Dyn. Syst.* 15, 2176–2212. doi: 10.1137/15M1052743

de Jong, H. (2002). Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* 9, 67–103. doi: 10.1089/10665270252833208

de Jong, H., Gouze, J., Hernandez, C., Page, M., Sari, T., and Geiselmann, J. (2004). Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bull. Math. Biol.* 66, 301–340. doi: 10.1016/j.bulm.2003.08.010

Edwards, R. (2001). Chaos in neural and gene networks with hard switching. *Diff. Eq. Dyn. Sys.* 9, 187–220.

Edwards, R., Machina, a., McGregor, G., and van den Driessche, P. (2015). A modelling framework for gene regulatory networks including transcription and translation. *Bull. Math. Biol.* 77, 953–983. doi: 10.1007/s11538-015-0073-9

Faure, A., Naldi, A., Chaouiya, C., , and Thieffry, D. (2006). Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics* 22, e124–e131. doi: 10.1093/bioinformatics/btl210

Glass, L., and Kauffman, S. a. (1972). Co-operative components, spatial localization and oscillatory cellular dynamics. *J. Theor. Biol.* 34, 219–37. doi: 10.1016/0022-5193(72)90157-9

Glass, L., and Kauffman, S. a. (1973). The logical analysis of continuous, non-linear biochemical control networks. *J. Theor. Biol.* 39, 103–129. doi: 10.1016/0022-5193(73)90208-7

Goncalves, E., Bucher, J., Ryll, A., Niklas, J., Mauch, K., Klamt, S., et al. (2013). Bridging the layers: towards integration of signal transduction, regulation and metabolism into mathematical models. *Mol. Biosyst.* 9, 1576–1583. doi: 10.1039/c3mb25489e

Harker, S. (2018). *DSGRN Software*. Available online at: https://github.com/shaunharker/DSGRN

Heatha, A., and Kavria, L. (2009). Computational challenges in systems biology. *Comput. Sci. Rev.* 3, 1–17. doi: 10.1016/j.cosrev.2009.01.002

Ironi, L., Panzeri, L., Plahte, E., and Simoncini, V. (2011). Dynamics of actively regulated gene networks. *Physica D* 240, 779–794. doi: 10.1016/j.physd.2010.12.010

Karlebach, G., and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature* 9:770. doi: 10.1038/nrm2503

Machado, D., Costa, R., Rocha, M., Ferreira, E., Tidor, B., and Rocha, I. (2011). Modeling formalisms in systems biology. *AMB Exp.* 1:45. doi: 10.1186/2191-0855-1-45

Pauleve, L., and Richard, A. (2012). Static analysis of boolean networks based on interaction graphs: a survey. *Electr. Notes Theor. Comput. Sci.* 284, 93–104. doi: 10.1016/j.entcs.2012.05.017

Saadatpour, A., and Reka, A. (2013). Boolean modeling of biological regulatory networks: a methodology tutorial. *Methods* 62, 3–12. doi: 10.1016/j.ymeth.2012.10.012

Thomas, R. (1973). Boolean formalization of genetic control circuits. *J. Theor. Biol.* 42, 563–585. doi: 10.1016/0022-5193(73)90247-6

Thomas, R., Thieffry, D., and Kaufman, M. (1995). Dynamical behaviour of biological regulatory networks-i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull. Math. Biol.* 57, 247–276. doi: 10.1007/BF02460618

Tournier, L., and Chaves, M. (2009). Uncovering operational interactions in genetic networks using asynchronous boolean dynamics. *J. Theor. Biol.* 260, 196–209. doi: 10.1016/j.jtbi.2009.06.006

Tyson, J. J., and Novak, B. (2013). "Chapter 14 - irreversible transitions, bistability and checkpoint controls in the eukaryotic cell cycle: a systems-level understanding," in *Handbook of Systems Biology*, ed A. M. W. V. Dekker (San Diego, CA: Academic Press), 265–285.

von Dassow, G., Meir, E., Munro, E., and Odell, G. (2000). The segment polarity network is a robust development module. *Nature* 406, 188–192. doi: 10.1038/35018085

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.